KB11-B

PDP11/70 MEM MGMT

CEKBECO

AH-7976C-MC

FICHE 1 OF 2

MAR 1979

digital

MADE IN USA

KB11-B

PDP11/70 MEM MGMT

CEKBECO

AH-7976C-MC

FICHE 2 OF 2

MAR 1979

digital

MADE IN USA

.REM @

IDENTIFICATION

PRODUCT CODE: AC-7975C-MC

PRODUCT NAME: CEKBEC0 11/70 MEM MGMT

PRODUCT DATE: 15-JAN-1979

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM WILL TEST ALL OF THE MEMORY MANAGEMENT LOGIC AND ENABLE THE FIELD SERVICE REPRESENTATIVE TO ISOLATE THE DETECTED FAILURES TO A REPLACABLE MODULE. IT IS ASSUMED THAT BOTH THE CPU AND THE CACHE HAVE BEEN TESTED, OR ARE KNOWN TO BE FUNCTIONING CORRECTLY, AND THAT THE PROGRAM IS STARTED FROM ADDRESS 200. THIS WILL PROVIDE THE EARLIEST DETECTION OF MEMORY MANAGEMENT RELATED ERRORS AND ENABLE LOOPING ON THE ERROR INVOLVING MINIMUM LOGIC. THIS PROGRAM MAY ALSO EXPOSE FAULTS THAT ARE ON THE INTERFACE BETWEEN MEMORY MANAGEMENT AND OTHER SECTIONS OF THE COMPUTER.

THIS PROGRAM HAS BEEN SEGMENTED IN THE FOLLOWING WAY:
ALL DATA TABLES, ERROR MESSAGES, AND SUBROUTINES RESIDE IN LOW CORE (VIRTUAL PAGES 0 & 1 IE. ADDRESSES 001100 THRU 037776). RIGHT NOW THE END OF THE SUBROUTINES IS AROUND 025000, SO THERE IS SOME ROOM FOR FUTURE EXPANSION. THE TEST CODE STARTS AT VIRTUAL PAGE 2 (ADDRESS 040000) AND EXPANDS TOWARD PAGE 4 (ADDRESS 100000). THE END OF THE PROGRAM IS NOW AROUND ADDRESS 074000, SO MODIFICATIONS CAN BE MADE WITHOUT RE-SEGMENTING THE PROGRAM.

THE REASON FOR THIS SEGMENTATION IS TWO-FOLD, FIRST IT ENABLES THE OPERATOR TO TELL FROM THE ADDRESS LIGHTS EXACTLY WHERE THE PROGRAM HAS HALTED OR "HUNG-UP". THAT IS, DID IT HALT IN THE ERROR ROUTINE OR IN A TRAP ROUTINE BECAUSE OF A CONDITION IMPOSSIBLE TO RECOVER FROM (ON PAGE 0 OR 1), OR DID IT GET "HUNG-UP" IN THE TEST CODE ON PAGE 2 OR 3.
THE OTHER REASON IS THAT CERTAIN MEMORY MANAGEMENT FUNCTIONS LOCK UP THE VIRTUAL PC OF THE INSTRUCTION AND THE PROGRAM, IN ORDER TO OPERATE PROPERLY, MUST KNOW WHERE IT IS AT ALL TIMES. IT SEEMS MUCH SIMPLER FOR THE CODE TO START AT A PREDETERMINED BOUNDARY SO THAT IF THE MESSAGES CHANGE OR A NEW SUBROUTINE IS ADDED THE PAGE THAT THE CODE IS ON WILL REMAIN THE SAME.

EACH TEST WILL SET THE LOOP ON ERROR POINTER ($LPERR) TO THE MINIMUM NECESSARY SETUP CODE, IF ANY, FOR THE FUNCTION UNDER TEST. A SYNCHRONIZATION INSTRUCTION (NOP) IS PROVIDED BEFORE THE INSTRUCTION(S) THAT TEST(S) EACH NEW FUNCTION. THIS WILL ENABLE THE FIELD SERVICE REPRESENTATIVE TO UTILIZE THE MICRO BREAK REGISTER TO GENERATE AN "EXTERNAL SYNC" PULSE ON THE BACK PLANE FOR BETTER PULSE RESOLUTION.

SECTION 8.2 OF THIS DOCUMENT CONTAINS SOME IDEAS THAT I HAD WHEN I WAS WRITING THIS PROGRAM ON HOW TO EFFECTIVELY UTILIZE IT TO MAKE FAULT ISOLATION EASIER. IF THESE IDEAS ARE NOT CORRECT OR NEED TO BE EXPANDED TO PROVIDE MORE INFORMATION PLEASE WRITE DOWN YOUR SUGGESTIONS AND FORWARD THEM TO THE DIAGNOSTIC DEPARTMENT.

IT SHOULD BE NOTED THAT THIS PROGRAM DOES NOT CHECK OUT THE

CONSOLE OR THE CONSOLE CABLES THAT PLUG INTO THE MEMORY MANAGEMENT BOARDS. THE PROGRAM ASSUMES THAT THOSE COMPONENTS HAVE BEEN TESTED OR ARE KNOWN TO BE GOOD.

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/70 COMPUTER, INCLUDING AN OPERATING CPU, CACHE, AND MEMORY. AN LA-30 OR EQUIVALENT DEVICE IS ALSO NEEDED FOR ERROR MESSAGES, AND END OF PASS REPORTS.

2.2 STORAGE

THIS PROGRAM REQUIRES 16K OF MEMORY TO LOAD AND AT LEAST 20K OF MEMORY TO RUN IN. IT WILL SCAN MEMORY FROM 16K TO 124K ON 2K BOUNDARIES, AND FROM 120K TO THE SIZE JUMPERS ON 8K BOUNDARIES.

2.3 PRELIMINARY PROGRAMS

THE CPU AND CACHE DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. MAIN MEMORY SHOULD BE SCANNED FOR AT LEAST THE FIRST 28K TO SEE THAT A PROGRAM WILL EXECUTE CORRECTLY BEFORE ANY PROGRAM IS RUN.

3. LOADING PROCEDURE

3.1 METHOD

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP, AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

200 THIS ADDRESS WILL RUN THE COMPLETE PROGRAM

204 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 2
TEST THE READ/WRITE BITSIN THE MEMORY STATUS REGISTERS

210 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 3
PAGE ADDRESS AND PAGE DESCRIPTOR TESTS

214 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 4
RELOCATION AND ADDER TESTS

220 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 5
MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS

224 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 6
D-SPACE TESTS, CORRECT TIMING OF I & D SPACE

230 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 7
A & W BIT LOGIC TEST AND DUAL MAPPING TESTS

234 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 8
MOVE FROM AND MOVE TO PERVIOUS MODE INSTRUCTION TESTS

4.2 PROGRAM AND OPERATOR ACTION

AFTER THE PROGRAM IS LOADED, THE FIRST TIME IT IS RUN IT WILL IDENTIFY ITSELF AND RUN A QUICK VERIFY PASS. AT THE END OF EACH PASS THE PROGRAM WILL TYPE OUT THE PASS NUMBER AND THE TOTAL NUMBER OF ERRORS FOUND ON THAT PASS.

4.3 SPECIAL STARTING PROCEDURE

IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN THIS PROGRAM WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL REGISTER (17777746) WITH THE DESIRED NUMBER, THEN LOAD THE PC (17777707) WITH THE STARTING ADDRESS AND PRESS CONTINUE. THE PROGRAM WILL NOW RUN AS IF YOU HAD LOADED THE STARTING ADDRESS AND PRESSED START BUT NOW THE CACHE CONTROL REGISTER IS DISABLING THE CACHE.

BIT00 -DISABLE TRAPS
BIT01 -DISABLE UNIBUS TRAPS
BIT02 -FORCE MISS ON READ GROUP 0
BIT03 -FORCE MISS ON READ GROUP 1
BIT04 -FORCE REPLACE GROUP 0
BIT05 -FORCE REPLACE GROUP 1

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SW15 1= HALT ON ERROR
SW14 1= LOOP ON THE TEST THAT YOU ARE IN
SW13 1= INHIBIT ALL ERROR TYPE OUTS
SW12 1= INHIBIT TRACE TRAP ON EVERY OTHER PASS
SW11 1= INHIBIT ITERATIONS AFTER FIRST PASS
SW10 1= RING BELL ON ERROR
SW09 1= LOOP ON ERROR
SW08 1= LOOP ON TEST IN SWR<06:00>
SW07 1= INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUBROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE, BEFORE THEIR EXPANSION, AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS. THE FOLLOWING IS A LIST OF THEIR TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

END OF PASS ROUTINE
SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR MESSAGE TYPE OUT ROUTINE
CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
SAVE & RESTORE R0-R5 ROUTINES
TYPE ROUTINE
BINARY TO OCTAL (ASCII) AND TYPE ROUTINE
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
TRAP DECODER
POWER DOWN AND UP ROUTINE
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

```
                        TURN OFF AND SAVE T-BIT
                        RESTORE T-BIT TO ITS PREVIOUS CONDITION
                        CLEAR 16 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
                        CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
                        P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
                        DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
                        COUNT PATTERN ERROR IN P.A.R.'S OR P.D.R.'S

                5.2.3   TRAP AND ABORT HANDLER ROUTINES

                        CPU TRAP HANDLER
                        CACHE TRAPS AND ABORTS HANDLER
                        MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER
                        TRAP ROUTINES FOR ABORT IN SUPERVISOR AND USER MODE


                6.      ERRORS
                        ------

                6.1     ERROR HALTS AND DESCRIPTION
                        WHEN THE PROGRAM DETECTS AN ERROR CONDITION IT ISSUES AN
                        ''ERROR'' (EMT) CALL.  THIS CAUSES THE CPU TO TRAP TO THE
                        ''ERROR HANDLER RUOTINE'' WHICH PRINTS OUT THE ERROR MESSAGE,
                        IF ANY, AND CHECKS THE SWITCH REGISTER FOR THE MODE SELECTED.
                        THE PROGRAM WILL REACT AS FOLLOWS:
                        HALT ON THE ERROR                IF SW15=1, IS UP
                        INHIBIT ERROR TYPE OUT           IF SW13=1, IS UP
                        RING BELL ON THE ERROR           IF SW10=1, IS UP
                        LOOP ON THE ERROR                IF SW09=1, IS UP
                        INHIBIT MULTIPLE TYPE OUTS       IF SW07=1, IS UP


                6.2     ERROR RECOVERY
                        IF SWITCH 09 IS UP, THE PROGRAM WILL LOOP BACK TO WHERE THE
                        LOOP ON ERROR POINTER ($LPERR) IS SET.  THIS WILL PROVIDE
                        THE TIGHTEST POSSIBLE SCOPING LOOP, AND PROVIDES ALL NECESSARY
                        SETUP CODE TO RECREATE THE ERROR.  IF A SYNC POINT IS DESIRED
                        TO EXTERNAL SYNC THE SCOPE JUST PRIOR TO THE FAILING OPERATION
                        LOAD THE MICRO BREAK REGISTER WITH ''044'' AND USE THE ''NOP''
                        PROVIDED.

                6.3     SAMPLE ERROR TYPE OUTS
CPU TRAP OR ABORT THRU 'ERRVEC' (004) HAD INCORRECT CONDITION
EXPECTD RECEIVD TESTNO PC AT ABORT
000040  000020  000047  XXXXXX

THIS ERROR MESSAGE INDICATES THAT THECPU TIMED OUT OVER THE
UNIBUS WHEN IT WAS EXPECTING A CACHE NON-EXISTANT MEMORY TRAP.
THIS TEST IS CHECKING THE CARRY PROPAGATION, THAT IS DETERMINED FROM
LOOKING AT THE INDEX AT THE FRONT OF THE LISTING.
```

```
                                        7.      RESTRICTIONS
                                                ------------

                                        7.1     STARTING RESTRICTIONS
                                                IF A STARTING POINT OTHER THAN '200'' IS USED AND ERRORS ARE
                                                REPORTED, THEY MAY BE DUE TO LOGIC THAT IS ASSUMED TO BE WORKING
                                                AS A RESULT OF TESTS THAT WERE NOT RUN.

                                        7.2     OPERATING RESTRICTIONS
                                                NONE

                                        8.      MISCELLANEOUS
                                                -------------

                                        8.1     EXECUTION TIME
                                                THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS IS
                                                APPROXIMATELY 10 SECONDS.

                                        8.2     HINTS ON HOW TO GET MORE INFORMATION FROM THE PROGRAM

                                                IF AN ERROR OCCURS THE FIRST THING THAT SHOULD BE NOTED IS
                                                WHAT PASS DID THE ERROR OCCUR ON.  IF THE PASS HAS AN EVEN
                                                NUMBER AND SWITCH 12 IS DOWN THEN THE ERROR MIGHT BE T-BIT
                                                SENSITIVE.  TRY TO RUN AGAIN WITH SWITCH 12 UP, THIS WILL
                                                INHIBIT T-BIT TRAPPING.
                                                IF THE PASS NUMBER IS GREATER THAN ONE THE ERROR MIGHT BE
                                                ITERATION SENSITIVE.  TRY TO RUN AGAIN WITH SWITCH 11 UP,
                                                THIS WILL INHIBIT ITERATIONS.
                                                NOW THAT YOU HAVE DETERMINED HOW TO MAKE THE MACHINE FAIL
                                                LOOK IN THE INDEX AT THE FRONT OF THE LISTING TO FIND THE TITLE
                                                OF THE TEST THAT WAS RUNNING WHEN THE ERROR CONDITION OCCURRED.
                                                GO TO THE LISTING AND READ THE PARAGRAPH AT THE BEGINNING OF
                                                THE TEST SO THAT YOU KNOW WHAT THE TEST IS TRYING TO DO.
                                                NOW READ THE ERROR MESSAGE AND IF THERE IS A COLUMN LABELED
                                                'ERRORPC'' GO TO THAT LOCATION IN THE TEST.  THIS IS THE PC
                                                OF THE 'ERROR'' STATEMENT.  THE NUMBER IS THE ERROR MESSAGE
                                                NUMBER AND IN THE FRONT OF THE LISTING IS THE 'ERROR MESSAGE
                                                POINTER TABLE'' WHICH WILL TELL YOU WHAT WORDS WERE TYPED
                                                OUT.
                                                IF YOU WANT TO SCOPE THIS ERROR CONDITION, PUT UP SWITCH 09
                                                (LOOP ON ERROR) OR IF YOU WANT TO LOOP ON THE ENTIRE TEST PUT
                                                UP SWITCH 14.  YOU WILL PROBABLY WANT TO INHIBIT THE ERROR TYPE
                                                OUT AT THIS POINT, SWITCH 13 WILL DO THAT.
                                                IF YOU NEED TO DO ACCURATE SCOPING AND NEED A GOOD SYNC POINT
                                                THEN MAKE SURE THAT THE MICRO BREAK REGISTER (1777770) HAS '044''
                                                IN IT.  THIS WILL CAUSE A PULSE ON THE BACK PLANE AT PIN #
                                                AE1 (SLOT 10) WHENEVER A 'NOP'' IS EXECUTED, AND THAT PULSE
                                                WILL MAKE AN EXCELLENT 'EXTERNAL SYNC'' SIGNAL FOR YOUR SCOPE.
                                                THERE IS A 'NOP'' JUST BEFORE EACH INSTRUCTION THAT TESTS A
                                                MEMORY MANAGEMENT FUNCTION FOR THE FIRST TIME, SO YOU SHOULD
                                                BE ABLE TO SCOPE ON ANY FAILURE THAT THIS PROGRAM CAN DETECT.
```

```
                                8.3     ROM STATE DESCRIPTIONS
                                        THIS IS A LIST OF THE ROM OUTPUTS FROM THE MEMORY MANAGEMENT
                                        ROMS ON 'SSRA', WITH A SENTENCE OR TWO DESCRIBING THEIR
                                        MEANING AS IT RELATES TO MEMORY MANAGEMENT.

                                        ROM OUT01 - ROM OUT03
                                        THESE OUTPUTS ARE SENT TO MULTIPLEXERS AND REGISTERS ON 'SSRA'
                                        TO INDICATE A PARTICULAR MACHINE FUNCTION.  THE ENCODING OF
                                        THESE OUTPUTS IS SHOWN IN A TRUTH TABLE ON 'SSRA'.

                                        ROM OUT04
                                        DESTINATION MODE -- THIS IS USED TO ENABLE RELOCATION IF BIT08
                                        OF MMR0 IS SET AND THIS IS THE DESTINATION CYCLE OF THE INSTRUCTION.

                                        ROM OUT05
                                        MFP + MTP -- THIS IS USED TO CLOCK THE PREVIOUS MODE INTO THE
                                        ''SPACE'' FLIP-FLOPS ON 'SSRB', DURING A MFP + MTP INST.

                                        ROM OUT06
                                        TRAP OR ABORT -- THIS IS USED TO FORCE SETTING OF THE
                                        'KERNEL SPACE' FLIP-FLOP ON 'SSRB' DURING A TRAP OR ABORT SEQUENCE.

                                        ROM OUT07
                                        BUST  -- THIS IS USED TO CLOCK MOST OF THE LATCHES IN MEMORY
                                        MANAGEMENT SUCH AS: 'SPACE' F/F'S, STATUS REGISTERS,...

                                        ROM OUT08
                                        MFP + MTP -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE I-SPACE ON
                                        MFPI + MTPI INSTRUCTIONS IF THE PSW IS NOT (USER MODE/
                                        PREVIOUS USER MODE).

                                        ROM OUT09
                                        INST + INDEX FETCH -- THIS ASSERTS 'SSRB I SPACEA' WHICH
                                        FORCES I-SPACE DURING AN INSTRUCTION OR INDEX WORD FETCH.

                                        ROM OUT10
                                        THIS DOES NOT EXIST.

                                        ROM OUT11
                                        INST STARTED IN I-SPACE -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE
                                        I-SPACE IF 'SSRB PREV=I' IS SET.

                                        ROM OUT12
                                        CONSOLE -- THIS ASSERTS 'SSRB I SPACEA' IF 'SSRK CNSL I SPACE H'
                                        IS TRUE AND YOU EXAMINE OR DEPOSIT.

                                        ROM OUT13
                                        SRCM = 1+2+3+4+5 -- IF THE SOURCE FIELD IS 7 THIS ASSERTS 'SSRB
                                        I SPACEB' WHICH FORCES I-SPACE.

                                        ROM OUT14
                                        DSTM = 1+2 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB
                                        I SPACEB' WHICH FORCES I-SPACE.
```

```
425
426
427                                     ROM OUT15
428                                     DSTM = 3 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB I
429                                     SPACEA' WHICH FORCES I-SPACE DURING THE DESTINATION CYCLE.
430
431                                     ROM OUT16
432                                     FLOATING POINT -- IF THE DSTF = 7 AND THE DSTM = 2 THEN THIS
433                                     ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE ON IMMEDIATE F.P.INST.
434
435                                     NOTE:   THE FOLLOWING ROM STATES ARE NOT EXPLICITLY TESTED DUE
436                                     TO THE FACT THAT I COULDN'T FIGURE OUT A WAY TO TEST THEM UNDER
437                                     RUN CONDITIONS.  THE LOGIC ASSOCIATED WITH THEM HAS BEEN TESTED
438                                     BY OTHER ROM STATES BUT THESE STATES ARE NOT TESTED.
439
440                                     ROM OUT05
441                                     SHR.00, NEG.00, D12.90, D40.30, D12.30
442
443                                     ROM OUT09
444                                     FET.06, FET.08, FET.09
445                                     THESE NEXT ARE TESTED ON PASSES WITH T-BIT TRAPPING
446                                     (PASS 2, 4, ...)
447                                     FET.01, FET.02, FET.03
448
449                                     ROM OUT12
450                                     EXM.10, EXM.20, DEP.10, DEP.20
451
452                                     ROM OUT13
453                                     S45.10
454
455                                     ROM OUT16   FLOATING POINT
456                                     FSV.00, FSV.10
457
458
459                             9.      PROGRAM DESCRIPTION
460                                     -------------------
461
462                                     THE DOCUMENT THAT FOLLOWS THIS ONE HAS A PARAGRAPH DESCRIBING
463                                     EACH OF THE TESTS.  THERE IS ALSO A PARAGRAPH DESCRIBING
464                                     EACH MAJOR GROUP OF TESTS.
465                             @
```

```
                        .TITLE  CEKBEC 11/70 MEM MGMT DIAGNOSTIC
                        ;*COPYRIGHT (C) JANUARY 15, 1979
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-A3).
                        ;*

                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH              USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            INHIBIT TRACE TRAP
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<6:0>
                        ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS

                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100            ;;FIRST ADDRESS OF THE STACK
001100                  KERSTK= STACK           ;;KERNEL STACK
000700                  SUPSTK= STACK-200       ;;SUPERVISOR STACK
000600                  USESTK= STACK-300       ;;USER STACK
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  SWR=    177570          ;;SWITCH REGISTER
177570                  DISPLAY=SWR

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE LINE FEED
000015                  CR=     15              ;;CODE CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
                        .EQUIV  R0,R10          ;;GENERAL REGISTER
```

```
                                .EQUIV  R1,R11          ;;GENERAL REGISTER
                                .EQUIV  R2,R12          ;;GENERAL REGISTER
                                .EQUIV  R3,R13          ;;GENERAL REGISTER
                                .EQUIV  R4,R14          ;;GENERAL REGISTER
                                .EQUIV  R5,R15          ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
                                .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
                                .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
                                .EQUIV  SP,USP          ;;USER STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER

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

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

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                          BIT15=  100000
040000                          BIT14=  40000
020000                          BIT13=  20000
010000                          BIT12=  10000
004000                          BIT11=  4000
002000                          BIT10=  2000
```

```
                        
000578  001000          BIT09=  1000
000579  000400          BIT08=  400
000580  000200          BIT07=  200
000581  000100          BIT06=  100
000582  000040          BIT05=  40
000583  000020          BIT04=  20
000584  000010          BIT03=  10
000585  000004          BIT02=  4
000586  000002          BIT01=  2
000587  000001          BIT00=  1
000588                  .EQUIV  BIT09,BIT9
000589                  .EQUIV  BIT08,BIT8
000590                  .EQUIV  BIT07,BIT7
000591                  .EQUIV  BIT06,BIT6
000592                  .EQUIV  BIT05,BIT5
000593                  .EQUIV  BIT04,BIT4
000594                  .EQUIV  BIT03,BIT3
000595                  .EQUIV  BIT02,BIT2
000596                  .EQUIV  BIT01,BIT1
000597                  .EQUIV  BIT00,BIT0
000598
000599                  ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
000600  000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000601  000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000602  000014          TBITVEC=14              ;;''T'' BIT
000603  000014          TRTVEC= 14              ;;TRACE TRAP
000604  000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000605  000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000606  000024          PWRVEC= 24              ;;POWER FAIL
000607  000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000608  000034          TRAPVEC=34              ;;''TRAP'' TRAP
000609  000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
000610  000064          TPVEC=  64              ;;TTY PRINTER VECTOR
000611  000114          CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
000612  000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
000613  000250          MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR
000614
000615                  .SBTTL  CACHE   REGISTER DEFINITIONS
000616
000617
000618  177740          LOADRS = 177740         ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
000619  177742          HIADRS = 177742         ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
000620  177744          MEMERR = 177744         ;;CACHE ERROR REGISTER
000621  177746          CONTRL = 177746         ;;MEMORY CONTROL REGISTER
000622  177750          MAINT  = 177750         ;;MEMORY MAINTENENCE REGISTER
000623  177752          HITMIS = 177752         ;;HIT MISS REGISTER ''1'' IMPLIES HIT IN CACHE
000624
000625
000626                  .SBTTL  CPU REGISTER DEFINITIONS
000627
000628
000629  177760          SIZELO = 177760         ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
000630                                          ;;TO GET TO THE LAST 32 WORDS OF MEMORY
000631  177762          SIZEHI = 177762         ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
000632                                          ;;CURRENTLY ALL ZERO
000633  177764          SYSTID = 177764         ;;SYSTEM ID REGISTER
```

```
                177766          CPUERR = 177766         ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
                                                        ;;THE TRAP TO ERRVEC (000004)



                                .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                                ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

                177572          MMR0=   177572
                177574          MMR1=   177574
                177576          MMR2=   177576
                172516          MMR3=   172516
                                .EQUIV  MMR0,SR0
                                .EQUIV  MMR1,SR1
                                .EQUIV  MMR2,SR2
                                .EQUIV  MMR3,SR3

                                ;*USER ''I'' PAGE DESCRIPTOR REGISTERS

                177600          UIPDR0= 177600
                177602          UIPDR1= 177602
                177604          UIPDR2= 177604
                177606          UIPDR3= 177606
                177610          UIPDR4= 177610
                177612          UIPDR5= 177612
                177614          UIPDR6= 177614
                177616          UIPDR7= 177616

                                ;*USER ''D'' PAGE DESCRIPTOR REGISTORS

                177620          UDPDR0= 177620
                177622          UDPDR1= 177622
                177624          UDPDR2= 177624
                177626          UDPDR3= 177626
                177630          UDPDR4= 177630
                177632          UDPDR5= 177632
                177634          UDPDR6= 177634
                177636          UDPDR7= 177636

                                ;*USER ''I'' PAGE ADDRESS REGISTERS

                177640          UIPAR0= 177640
                177642          UIPAR1= 177642
                177644          UIPAR2= 177644
                177646          UIPAR3= 177646
                177650          UIPAR4= 177650
                177652          UIPAR5= 177652
                177654          UIPAR6= 177654
                177656          UIPAR7= 177656

                                ;*USER ''D'' PAGE ADDRESS REGISTERS

                177660          UDPAR0= 177660
```

```
                177662          UDPAR1= 177662
                177664          UDPAR2= 177664
                177666          UDPAR3= 177666
                177670          UDPAR4= 177670
                177672          UDPAR5= 177672
                177674          UDPAR6= 177674
                177676          UDPAR7= 177676

                                ;*SUPERVISOR ''I'' PAGE DESCRIPTOR REGISTERS

                172200          SIPDR0= 172200
                172202          SIPDR1= 172202
                172204          SIPDR2= 172204
                172206          SIPDR3= 172206
                172210          SIPDR4= 172210
                172212          SIPDR5= 172212
                172214          SIPDR6= 172214
                172216          SIPDR7= 172216

                                ;*SUPERVISOR ''D'' PAGE DESCRIPTOR REGISTERS

                172220          SDPDR0= 172220
                172222          SDPDR1= 172222
                172224          SDPDR2= 172224
                172226          SDPDR3= 172226
                172230          SDPDR4= 172230
                172232          SDPDR5= 172232
                172234          SDPDR6= 172234
                172236          SDPDR7= 172236

                                ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS

                172240          SIPAR0= 172240
                172242          SIPAR1= 172242
                172244          SIPAR2= 172244
                172246          SIPAR3= 172246
                172250          SIPAR4= 172250
                172252          SIPAR5= 172252
                172254          SIPAR6= 172254
                172256          SIPAR7= 172256

                                ;*SUPERVISOR ''D'' PAGE ADDRESS REGISTERS

                172260          SDPAR0= 172260
                172262          SDPAR1= 172262
                172264          SDPAR2= 172264
                172266          SDPAR3= 172266
                172270          SDPAR4= 172270
                172272          SDPAR5= 172272
                172274          SDPAR6= 172274
                172276          SDPAR7= 172276

                                ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS

                172300          KIPDR0= 172300
                172302          KIPDR1= 172302
```

```
                        172304          KIPDR2= 172304
                        172306          KIPDR3= 172306
                        172310          KIPDR4= 172310
                        172312          KIPDR5= 172312
                        172314          KIPDR6= 172314
                        172316          KIPDR7= 172316

                                        ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

                        172320          KDPDR0= 172320
                        172322          KDPDR1= 172322
                        172324          KDPDR2= 172324
                        172326          KDPDR3= 172326
                        172330          KDPDR4= 172330
                        172332          KDPDR5= 172332
                        172334          KDPDR6= 172334
                        172336          KDPDR7= 172336

                                        ;*KERNEL 'I' PAGE ADDRESS REGISTERS

                        172340          KIPAR0= 172340
                        172342          KIPAR1= 172342
                        172344          KIPAR2= 172344
                        172346          KIPAR3= 172346
                        172350          KIPAR4= 172350
                        172352          KIPAR5= 172352
                        172354          KIPAR6= 172354
                        172356          KIPAR7= 172356

                                        ;*KERNEL 'D' PAGE ADDRESS REGISTERS

                        172360          KDPAR0= 172360
                        172362          KDPAR1= 172362
                        172364          KDPAR2= 172364
                        172366          KDPAR3= 172366
                        172370          KDPAR4= 172370
                        172372          KDPAR5= 172372
                        172374          KDPAR6= 172374
                        172376          KDPAR7= 172376



                                        .SBTTL  UNIBUS MAP REGISTER DEFINITIONS


                                        ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                        ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'


                        170200          MAPL00 = 170200
                        170202          MAPH00 = 170202
                        170204          MAPL01 = 170204
                        170206          MAPH01 = 170206
                        170210          MAPL02 = 170210
                        170212          MAPH02 = 170212
```

```
                                                    D 2
  802          170214          MAPL03 = 170214
  803          170216          MAPH03 = 170216
  804          170220          MAPL04 = 170220
  805          170222          MAPH04 = 170222
  806          170224          MAPL05 = 170224
  807          170226          MAPH05 = 170226
  808          170230          MAPL06 = 170230
  809          170232          MAPH06 = 170232
  810          170234          MAPL07 = 170234
  811          170236          MAPH07 = 170236
  812          170240          MAPL10 = 170240
  813          170242          MAPH10 = 170242
  814          170244          MAPL11 = 170244
  815          170246          MAPH11 = 170246
  816          170250          MAPL12 = 170250
  817          170252          MAPH12 = 170252
  818          170254          MAPL13 = 170254
  819          170256          MAPH13 = 170256
  820          170260          MAPL14 = 170260
  821          170262          MAPH14 = 170262
  822          170264          MAPL15 = 170264
  823          170266          MAPH15 = 170266
  824          170270          MAPL16 = 170270
  825          170272          MAPH16 = 170272
  826          170274          MAPL17 = 170274
  827          170276          MAPH17 = 170276
  828          170300          MAPL20 = 170300
  829          170302          MAPH20 = 170302
  830          170304          MAPL21 = 170304
  831          170306          MAPH21 = 170306
  832          170310          MAPL22 = 170310
  833          170312          MAPH22 = 170312
  834          170314          MAPL23 = 170314
  835          170316          MAPH23 = 170316
  836          170320          MAPL24 = 170320
  837          170320          MAPH24 = 170320
  838          170324          MAPL25 = 170324
  839          170326          MAPH25 = 170326
  840          170330          MAPL26 = 170330
  841          170332          MAPH26 = 170332
  842          170334          MAPL27 = 170334
  843          170336          MAPH27 = 170336
  844          170340          MAPL30 = 170340
  845          170342          MAPH30 = 170342
  846          170344          MAPL31 = 170344
  847          170346          MAPH31 = 170346
  848          170350          MAPL32 = 170350
  849          170352          MAPH32 = 170352
  850          170354          MAPL33 = 170354
  851          170356          MAPH33 = 170356
  852          170360          MAPL34 = 170360
  853          170362          MAPH34 = 170362
  854          170364          MAPL35 = 170364
  855          170366          MAPH35 = 170366
  856          170370          MAPL36 = 170370
  857          170372          MAPH36 = 170372
```

```
                170374                  MAPL37 = 170374
                170376                  MAPH37 = 170376
                                        .EQUIV  MAPL00,MAPL0
                                        .EQUIV  MAPH00,MAPH0
                                        .EQUIV  MAPL01,MAPL1
                                        .EQUIV  MAPH01,MAPH1
                                        .EQUIV  MAPL02,MAPL2
                                        .EQUIV  MAPH02,MAPH2
                                        .EQUIV  MAPL03,MAPL3
                                        .EQUIV  MAPH03,MAPH3
                                        .EQUIV  MAPL04,MAPL4
                                        .EQUIV  MAPH04,MAPH4
                                        .EQUIV  MAPL05,MAPL5
                                        .EQUIV  MAPH05,MAPH5
                                        .EQUIV  MAPL06,MAPL6
                                        .EQUIV  MAPH06,MAPH6
                                        .EQUIV  MAPL07,MAPL7
                                        .EQUIV  MAPH07,MAPH7

                                        .SBTTL  TRAP CATCHER

                000000                          .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

                                        .SBTTL  STARTING ADDRESS(ES)
                000200                          .=200

        000200  000137  040000                  JMP     @#STRT1         ;;JUMP TO STARTING ADDRESS OF PROGRAM
                                                                        ;RUN ENTIRE PROGRAM (ALL TESTS)
        000204  000137  040010                  JMP     STRT2           ;STARTING AT ENTRY POINT 2
                                                                        ;MEMORY MANAGEMENT STATUS REGISTERS
        000210  000137  040020                  JMP     STRT3           ;STARTING AT ENTRY POINT 3
                                                                        ;PAGE ADDRESS AND DESCRIPTOR REGISTERS
        000214  000137  040030                  JMP     STRT4           ;STARTING AT ENTRY POINT 4
                                                                        ;RELOCATION AND ADDER TESTS
        000220  000137  040040                  JMP     STRT5           ;STARTING AT ENTRY POINT 5
                                                                        ;MEMORY MANAGEMENT TRAP AND ABORT LOGIC
        000224  000137  040050                  JMP     STRT6           ;STARTING AT ENTRY POINT 6
                                                                        ;D-SPACE ENABLING
        000230  000137  040060                  JMP     STRT7           ;STARTING AT ENTRY POINT 7
                                                                        ;A & W BIT LOGIC & DUAL MAPPING
        000234  000137  040070                  JMP     STRT8           ;STARTING AT ENTRY POINT 8
                                                                        ;MFP & MTP LOGIC TESTS
                                        ;;*****************************************************************

                                        .SBTTL          ACT11 HOOKS

                                        ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
                                        ;*
                                        ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOCICAL
```

```
                                ;*END OF THE PROGRAM.
                                ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
                                ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
                                ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
                                ;*
                                ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
                                ;*            =0 NO POWER FAIL DESIRED
                                ;*
                                ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
                                ;*            =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
                                ;*
                                ;*      BITS 13-0  MUST BE ZERO'S

            000240              $SVPC=.                 ;;SAVE LOCATION COUNTER
            000046              .=46                    ;;SET LOCATION COUNTER
000046      021230              .WORD   $ENDAD          ;;SET LOC.46 TO ADDRESS $ENDAD
            000052              .=52                    ;;SET LOCATION COUNTER
000052      000000              .WORD   0               ;;SET LOC.52 TO ZERO
            000240              .=$SVPC                 ;; RESTORE LOCATION COUNTER
```

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

                        .SBTTL  COMMON TAGS

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

          001100                .=1100

001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102     000          $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000          $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000          $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001          $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS  OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS  OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000  000000  000000          .WORD   0,0,0   ;;RESERVED--NOT TO BE USED
001136  177560          $TKS:   177560                  ;;TTY KBD STATUS
001140  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001142  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG.
001144  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG.
001146     000          $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001147     002          $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001150     012          $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
001151     000          $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001152  000000          $REGAD: .WORD   0               ;;CONTAINS THE  FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001154  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001156  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001160  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001162  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001164  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001166  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001170  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001172  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001174  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001176  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001200  000000          $TMP4:  .WORD   0               ;;USER DEFINED
001202  000000          $TMP5:  .WORD   0               ;;USER DEFINED
001204  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001206  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR
001210  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001214     077          $QUES:  .ASCII  /?/             ;;QUESTION MARK
001215     015          $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001216  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
001220  000000          FSTTST: .WORD   0               ;HOLDS THE INDEX TO
                                                        ;THE STARTING ADDRESS TABLE
001222  000000          CPUEXP: .WORD   0               ;HOLDS EXPECTED CPU ERROR CONDITION
```

```
001224  000000          MMEXP:  .WORD               ;HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION
001226  000000          PADRSL: .WORD   0           ;HOLDS THE LOWER 16 BITS OF A 22-BIT
                                                    ;ADDRESS GENERATED FOR TYPE OUT
001230  000000          PADRSH: .WORD   0           ;HOLDS THE UPPER 6 BITS OF A 22-BIT
                                                    ;ADDRESS GENERATED FOR TYPE OUT
001232  000000          PLOADR: .WORD   0           ;HOLDS CACHE LO ADDR REG
001234  000000          PHIADR: .WORD   0           ;HOLDS CACHE HI ADDR REG
001236  000000          PPARER: .WORD   0           ;HOLDS PARITY ERROR REG
001240  000000          PCONTR: .WORD   0           ;HOLDS CACHE CONTROL REG
001242  000000          PMAINT: .WORD   0           ;HOLDS CACHE MAINTENENCE REG
001244  000000          PHITMI: .WORD   0           ;HOLDS CACHE HIT MISS REG
001246  000000          PMMR0:  .WORD   0           ;HOLDS MEMORY MANAGEMENT REGISTER 0
001250  000000          PMMR1:  .WORD   0           ;HOLDS MEMORY MANAGEMENT REGISTER 1
001252  000000          PMMR2:  .WORD   0           ;HOLDS MEMORY MANAGEMENT REGISTER 2
001254  000000          PMMR3:  .WORD   0           ;HOLDS MEMORY MANAGEMENT REGISTER 3
001256  000000          PCPUER: .WORD   0           ;HOLDS CPU ERROR REGISTER.
001260  000000          BADPC:  .WORD   0           ;HOLDS PC AT ABORT OR TRAP TIME.
001262  000000          TESTNO: .WORD   0           ;HOLDS TEST NUMBER OF LAST ERROR.
001264  000000          DATAOR: .WORD   0           ;HOLDS LOGICAL OR OF BAD DATA
001266  000000          DATAND: .WORD   0           ;HOLDS LOGICAL AND OF BAD DATA
001270  000000          PATTOR: .WORD   0           ;HOLDS LOGICAL OR OF PATTERN LOADED
001272  000000          PATAND: .WORD   0           ;HOLDS LOGICAL AND OF PATTERN LOADED
001274  000000          ADDROR: .WORD   0           ;HOLDS LOGICAL OR OF ADDRESS
001276  000000          ADRAND: .WORD   0           ;HOLDS LOGICAL AND OF ADDRESS
001300  000000          ERRCNT: .WORD   0           ;HOLDS NUMBER OF ERRORS ON TEST
001302  000000          HOLFLG: .WORD   0           ;HOLDS NUMBER OF CONSECUTIVE TIME OUTS
                                                    ;OCCURRING IN A HOLE IN MEMORY
001304  000000          OLDPC:  .WORD   0           ;HOLDS THE RETURN ADDRESS
                                                    ;IN CASE OF A LOOP ON ERROR
001306  000000          OLDPS:  .WORD   0           ;HOLDS THE OLD PROCESSOR STATUS
                                                    ;IN CASE OF A LOOP ON ERROR
001310  000340          OLDPSW: .WORD   000340      ;HOLDS THE OLD PSW SO THAT THE T-BIT
                                                    ;CAN BE RESTORED PROPERLY
001312  000000          RETRY:  .WORD   0           ;FLAG TO DECIDE IF ONE PARITY HAS
                                                    ;HAS BEEN ATTEMPTED
001314  000000          NXTTST: .WORD   0           ;HOLDS ADDRESS OF THE NEXT TEST

                        ;:  THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL
                        ;:  REFER TO.  ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT
                        ;:  IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.

001316  020000          READON: .WORD   20000       ;READ ONLY BIT IN MMR0


001320                  PARTAB:                     ;THIS IS THE TABLE OF THE FIRST
                                                    ;PAR OR PDR OF EACH GROUP.  THEY
                                                    ;WILL BE USED FOR A DUAL ADDRESSING
                                                    ;TEST BETWEEN GROUPS.
001320  172200                  .WORD   172200      ;SIPDR0
001322  172240                  .WORD   172240      ;SIPAR0
001324  172300                  .WORD   172300      ;KIPDR0
001326  172340                  .WORD   172340      ;KIPAR0
001330  177600                  .WORD   177600      ;UIPDR0
001332  177640                  .WORD   177640      ;UIPAR0
001334  040536          STRTAB: .WORD   TST1        ;STARTING ADDRESS OF TEST ONE
001336  041436                  .WORD   ENTPT2      ;ADDRESS OF ENTRY POINT 2
```

```
1045  001340  042614                .WORD   ENTPT3          ;ADDRESS OF ENTRY POINT 3
1046  001342  047236                .WORD   ENTPT4          ;ADDRESS OF ENTRY POINT 4
1047  001344  052274                .WORD   ENTPT5          ;ADDRESS OF ENTRY POINT 5
1048  001346  057446                .WORD   ENTPT6          ;ADDRESS OF ENTRY POINT 6
1049  001350  062136                .WORD   ENTPT7          ;ADDRESS OF ENTRY POINT 7
1050  001352  067636                .WORD   ENTPT8          ;ADDRESS OF ENTRY POINT 8
1051  001354  001000      ENMMTR:   .WORD   BIT9            ;ENABLE MEMORY MANAGEMENT TRAPS  BIT
```

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

                                .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


001356                          $ERRTB:


                                ;ITEM 1
001356  002736                  EM1                     ;NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)
001360  013214                  DH1                     ;EXPECTD RECEIVD TESTNO  PC AT ABORT
001362  017416                  DT1                     ;CPUEXP,PCPUER,TESTNO,BADPC,0
001364  020466                  DF1                     ;0,0,0,0

                                ;ITEM 2
001366  003026                  EM2                     ;UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
001370  013224                  DH2                     ;RECEIVD TESTNO  PC AT ABORT
001372  017420                  DT2                     ;PCPUER,TESTNO,BADPC,0
001374  020467                  DF2                     ;0,0,0

                                ;ITEM 3
001376  003107                  EM3                     ;UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)
001400  013260                  DH3                     ;PARITY  ADDRESS   CONTROL MAINTEN
                                                        ;RECEIVD REFERENCD REGISTR REGISTR TESTNO  PC AT ABORT
001402  017430                  DT3                     ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0
001404  020472                  DF3                     ;0,2,0,0,0,0

                                ;ITEM 4
001406  003225                  EM4                     ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
001410  013260                  DH3                     ;PARITY  ADDRESS   CONTROL MAINTEN
                                                        ;RECEIVD REFERENCD REGISTR REGISTR TESTNO  PC AT ABORT
001412  017430                  DT3                     ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0
001414  020472                  DF3                     ;0,2,0,0,0,0

                                ;ITEM 5
001416  003350                  EM5                     ;UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT
001420  013410                  DH5                     ;ERROR   AUTOI/D VIRTUAL
                                                        ;REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT
001422  017446                  DT5                     ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
001424  020500                  DF5                     ;0,0,0,0,0

                                ;ITEM 6
001426  003423                  EM6                     ;MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION
001430  013514                  DH6                     ;EXPECTD ERROR   AUTOI/D VIRTUAL
                                                        ;CONDITN REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT
```

```
1108  001432  017462          DT6             ;MMEXP,PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
1109  001434  020505          DF6             ;0,0,0,0,0,0
1110
1111                          ;ITEM 7
1112  001436  003513          EM7             ;MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
1113  001440  013640          DH7             ;(MMR0)  TESTNO  ERRORPC
1114  001442  017500          DT7             ;$REG1,TESTNO,$ERRPC,0
1115  001444  020513          DF7             ;0,0,0
1116
1117                          ;ITEM 10
1118  001446  003567          EM10            ;CAN'T SET 171000 INTO MMR0
1119  001450  013640          DH7             ;(MMR0)  TESTNO  ERRORPC
1120  001452  017500          DT7             ;$REG1,TESTNO,$ERRPC,0
1121  001454  020513          DF7             ;0,0,0
1122
1123                          ;ITEM 11
1124  001456  003620          EM11            ;GOT THE WRONG DATA BACK FROM MMR0
1125  001460  013670          DH11            ;LOADED  RECEIVD TESTNO  ERRORPC
1126  001462  017510          DT11            ;$REG2,$REG1,TESTNO,$ERRPC,0
1127  001464  020516          DF11            ;0,0,0,0
1128
1129                          ;ITEM 12
1130  001466  003662          EM12            ;MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR
1131  001470  013740          DH12            ;(MMR1)  TESTNO  ERRORPC
1132  001472  017522          DT12            ;$REG2,TESTNO,$ERRPC,0
1133  001474  020522          DF12            ;0,0,0
1134
1135                          ;ITEM 13
1136  001476  003736          EM13            ;MMR1 DID NOT TRACK PROPERLY
1137  001500  013730          DH13            ;EXPECTD (MMR1)  TESTNO  ERRORPC
1138  001502  017532          DT13            ;$REG3,$REG2,TESTNO,$ERRPC,0
1139  001504  020525          DF13            ;0,0,0,0
1140
1141                          ;ITEM 14
1142  001506  003772          EM14            ;MMR2 DID NOT TRACK PROPERLY
1143  001510  013770          DH14            ;EXPECTD (MMR2)  TESTNO  ERRORPC
1144  001512  017532          DT13            ;$REG3,$REG2,TESTNO,$ERRPC
1145  001514  020525          DF13            ;0,0,0,0
1146
1147                          ;ITEM 15
1148  001516  004026          EM15            ;MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR
1149  001520  014040          DH15            ;(MMR3)  TESTNO  ERRORPC
1150  001522  017522          DT12            ;$REG,TESTNO,$ERRPC,0
1151  001524  020522          DF12            ;0,0,0
1152
1153                          ;ITEM 16
1154  001526  004102          EM16            ;MMR3 IS HOLDING THE WRONG DATA
1155  001530  014030          DH16            ;LOADED (MMR3)  TESTNO  ERRORPC
1156  001532  017544          DT16            ;$REG1,$REG2,TESTNO,$ERRPC,0
1157  001534  020531          DF16            ;0,0,0,0
1158
1159                          ;ITEM 17
1160  001536  004135          EM17            ;SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
1161  001540  014070          DH17            ;ADDROR  ADDRAND TESTNO  #ERRORS
1162  001542  017556          DT17            ;ADDROR,ADRAND,TESTNO,ERRCNT,0
1163  001544  020540          DF17            ;0,0,0,1
```

```
1164
1165                                    ;ITEM 20
1166  001546  004203                   EM20            ;FOLLOWING PAR/PDR WILL NOT ZERO
1167  001550  014130                   DH20            ;ADDRESS DATA    TESTNO  ERRORPC
1168  001552  017570                   DT20            ;$REG0,$REG2,TESTNO,$ERRPC,0
1169  001554  020544                   DF20            ;0,0,0,0
1170
1171                                    ;ITEM 21
1172  001556  004244                   EM21            ;SUMMARY OF DUAL ADDRESSING ERRORS
1173  001560  014170                   DH21            ;ADDROR  ADDRAND ADDROR  ADDRAND
1174                                                    ;LOADED  LOADED ENABLED ENABLED TESTNO  #ERRORS
1175  001562  017602                   DT21            ;ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0
1176  001564  020550                   DF21            ;0,0,0,0,0,1
1177
1178                                    ;ITEM 22
1179  001566  004306                   EM22            ;SUMMARY OF COUNT PATTERN FAILURES
1180  001570  014310                   DH22            ;ADDROR  ADDRAND PATRNOR PATRNAD DATAOR  DATAAND TESTNO  #ERRORS
1181  001572  017620                   DT22            ;ADDROR ADRAND,PATTOR,PATAMD,DATAOR,DATAND,TESTNO,ERRCNT,0
1182  001574  020556                   DF22            ;0,0,0,0,0,0,0,1
1183
1184                                    ;ITEM 23
1185  001576  004350                   EM23            ;ERROR IN BYTE ADDRESSING OF PAR/PDR
1186  001600  014407                   DH23            ;ADDRESS EXPECTD RECEIVD TESTNO  ERRORPC
1187  001602  017642                   DT23            ;$REG0,$REG2,$REG1,TESTNO,$ERRPC,0
1188  001604  020566                   DF23            ;0,0,0,0,0
1189
1190                                    ;ITEM 24
1191  001606  004414                   EM24            ;ONE OF THE REGISTERS TIMED OUT
1192  001610  014457                   DH24            ;REGADDR TESTNO  ERRORPC
1193  001612  017656                   DT24            ;$REG0,TESTNO,$ERRPC,0
1194  001614  020573                   DF24            ;0,0,0
1195
1196                                    ;ITEM 25
1197  001616  004453                   EM25            ;LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES
1198  001620  014507                   DH25            ;REGADDR DATA    TESTNO  ERRORPC
1199  001622  017666                   DT25            ;$REG0,$REG1,TESTNO,$ERRPC,0
1200  001624  020576                   DF25            ;0,0,0,0
1201
1202                                    ;ITEM 26
1203  001626  004531                   EM26            ;COULD WRITE ONE OF THE SIZE REGISTERS
1204  001630  014507                   DH25            ;REGADDR DATA    TESTNO  ERRORPC
1205  001632  017666                   DT25            ;$REG0,$REG2,TESTNO,$ERRPC,0
1206  001634  020576                   DF25            ;0,0,0,0
1207
1208                                    ;ITEM 27
1209  001636  004577                   EM27            ;HIGH SIZE REGISTER IS NOT ZERO
1210  001640  014507                   DH25            ;REGADDR DATA    TESTNO  ERRORPC
1211  001642  017666                   DT25            ;$REG0,$REG1,TESTNO,$ERRPC,0
1212  001644  020576                   DF25            ;0,0,0,0
1213
1214                                    ;ITEM 30
1215  001646  004636                   EM30            ;CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE.
1216  001650  014507                   DH25            ;REGADDR DATA    TESTNO  ERRORPC
1217  001652  017666                   DT25            ;$REG0,$REG1,TESTNO,$ERRPC
1218  001654  020576                   DF25            ;0,0,0,0
1219
```

```
1220                                    ;ITEM 31
1221  001656  004725                    EM31            ;LOWER BYTE OF P.S.W. NOT CORRECT
1222  001660  014547                    DH31            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
1223  001662  017700                    DT31            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1224  001664  020602                    DF31            ;0,0,0,0,0
1225
1226                                    ;ITEM 32
1227  001666  004766                    EM32            ;LOADED MORE THAN LOWER BYTE OF MICRO BREAK REGISTER
1228  001670  014547                    DH31            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
1229  001672  017700                    DT31            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1230  001674  020602                    DF31            ;0,0,0,0,0
1231
1232                                    ;ITEM 33
1233  001676  005052                    EM33            ;DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER
1234  001700  014547                    DH31            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
1235  001702  017700                    DT31            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1236  001704  020602                    DF31            ;0,0,0,0,0
1237
1238                                    ;ITEM 34
1239  001706  005131                    EM34            ;LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
1240  001710  014547                    DH31            ;REGADDR DATAREC DATAEXP TESTNO  ERRORPC
1241  001712  017700                    DT31            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1242  001714  020602                    DF31            ;0,0,0,0,0
1243
1244                                    ;ITEM 35
1245  001716  005215                    EM35            ;KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP AND USP
1246  001720  014617                    DH35            ;KSP     TESTNO  ERRORPC
1247  001722  017714                    DT35            ;$REG0,TESTNO,$ERRPC,0
1248  001724  020607                    DF35            ;0,0,0
1249
1250                                    ;ITEM 36
1251  001726  005303                    EM36            ;DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
1252  001730  014647                    DH36            ;INDEX   INDEX   PAR/PDR
1253                                                    ;EXPECTD RECEIVD ADRREAD TESTNO  ERRORPC
1254  001732  017724                    DT36            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1255  001734  020612                    DF36            ;0,0,0,0,0
1256
1257                                    ;ITEM 37
1258  001736  005352                    EM37            ;BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
1259  001740  014747                    DH37            ;ADDRESS GDDATA  BADDATA TESTNO  ERRORPC
1260  001742  017740                    DT37            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1261  001744  020617                    DF37            ;3,0,0,0,0
1262
1263                                    ;ITEM 40
1264  001746  005431                    EM40            ;18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1265  001750  015017                    DH40            ;STARTADR  FINISHADR TESTNO  ERRORPC
1266  001752  017754                    DT40            ;$TMP1,$TMP2,TESTNO,$ERRPC,0
1267  001754  020624                    DF40            ;4,4,0,0
1268
1269                                    ;ITEM 41
1270  001756  005512                    EM41            ;18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
1271  001760  015063                    DH41            ;STARTADR  TESTNO  ERRORPC
1272  001762  017766                    DT41            ;$TMP1,TESTNO,$ERRPC,0
1273  001764  020630                    DF41            ;4,0,0
1274
1275                                    ;ITEM 42
```

```
1276  001766  005574                EM42            ;FAULTY CARRY PROPAGATION 18-BIT MAPPING.
1277  001770  015115                DH42            ;PATTERN DATA    ADDRESS
1278                                                ;LOADED  FETCHED INTENDED  TESTNO  ERRORPC
1279  001772  017776                DT42            ;$REG2,$REG3,$REG0,TESTNO,$ERRPC,0
1280  001774  020633                DF42            ;0,0,3,0,0
1281
1282                                ;ITEM 43
1283  001776  005645                EM43            ;NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000
1284  002000  015217                DH43            ;TESTNO  ERRORPC
1285  002002  020012                DT43            ;TESTNO,$ERRPC,0
1286  002004  020640                DF43            ;0,0
1287
1288                                ;ITEM 44
1289  002006  005723                EM44            ;DIDN'T GET WRAP AROUND TO ADDRESS ZERO
1290  002010  015237                DH44            ;DATA    TESTNO  ERRORPC
1291  002012  020020                DT44            ;$REG1,TESTNO,$ERRPC,0
1292  002014  020642                DF44            ;0,0,0
1293
1294                                ;ITEM 45
1295  002016  005772                EM45            ;NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS
1296  002020  015267                DH45            ;NON-EXADDR TESTNO  ERRORPC
1297  002022  020030                DT45            ;$REG0,TESTNO,$ERRPC,0
1298  002024  020645                DF45            ;3,0,0
1299
1300                                ;ITEM 46
1301  002026  006047                EM46            ;BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING
1302  002030  015115                DH42            ;PATTERN DATA    ADDRESS
1303                                                ;LOADED  FETCHED INTENDED  TESTNO  ERRORPC
1304  002032  017776                DT42            ;$REG2,$REG3,$REG0,TESTNO,$ERRPC
1305  002034  020633                DF42            ;0,0,3,0,0
1306
1307                                ;ITEM 47
1308  002036  006130                EM47            ;DID NOT GET UNIBUS ADDRESS
1309  002040  015115                DH42            ;PATTERN DATA    ADDRESS
1310                                                ;LOADED  FETCHED INTENDED  TESTNO  ERRORPC
1311  002042  017776                DT42            ;$REG2,$REG3,$REG0,TESTNO,$ERRPC,0
1312  002044  020633                DF42            ;0,0,3,0,0
1313
1314                                ;ITEM 50
1315  002046  006163                EM50            ;COMPARE CIRCUIT FOR TOP OF MEMORY BAD
1316  002050  015322                DH50            ;KIPAR4    SIZELO    TESTNO  ERRORPC
1317  002052  020040                DT50            ;KIPAR4,SIZELO,TESTNO,$ERRPC,0
1318  002054  020650                DF50            ;4,4,0,0
1319
1320                                ;ITEM 51
1321  002056  006231                EM51            ;PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
1322  002060  015366                DH51            ;PGLENFD VABLKNO TESTNO  ERRORPC
1323  002062  020052                DT51            ;$REG1,$REG3,TESTNO,$ERRPC,0
1324  002064  020654                DF51            ;0,0,0,0
1325
1326                                ;ITEM 52
1327  002066  006304                EM52            ;NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
1328  002070  015366                DH51            ;PGLENFD VABLKNO TESTNO  ERRORPC
1329  002072  020052                DT51            ;$REG1,$REG3,TESTNO,$ERRPC,0
1330  002074  020654                DF51            ;0,0,0,0
1331
```

```
1332
1333  002076  006361                ;ITEM 53
1334  002100  015426                EM53    ;DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5
1335  002102  020064                DH53    ;TESTNO  ERRORPC
1336  002104  020660                DT53    ;TESTNO,$ERRPC,0
1337                                DF53    ;0,0
1338
1339  002106  006433                ;ITEM 54
1340  002110  015426                EM54    ;DID NOT ABORT ON READ ONLY PAGE KIPDR4
1341  002112  020064                DH53    ;TESTNO  ERRORPC
1342  002114  020660                DT53    ;TESTNO,$ERRPC,0
1343                                DF53    ;0,0
1344
1345  002116  006502                ;ITEM 55
1346  002120  015446                EM55    ;INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMR0) CLEAR
1347  002122  020072                DH55    ;EXPDATA RECDATA TESTNO  ERRORPC
1348  002124  020662                DT55    ;$REG0,$REG1,TESTNO,$ERRPC,0
1349                                DF55    ;0,0,0,0
1350
1351  002126  006565                ;ITEM 56
1352  002130  015506                EM56    ;NO M.M. TRAP WHEN BIT09 (MMR0) SET PAGE KIPDR4
1353  002132  020104                DH56    ;(MMR0)  KIPDR4  TESTNO  ERRORPC
1354  002134  020666                DT56    ;MMR0,KIPDR4,TESTNO,$ERRPC,0
1355                                DF56    ;0,0,0,0
1356
1357  002136  006644                ;ITEM 57
1358  002140  015446                EM57    ;INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMR0) SET
1359  002142  020072                DH55    ;EXPDATA RECDATA TESTNO  ERRORPC
1360  002144  020662                DT55    ;$REG0,$REG1,TESTNO,$ERRPC,0
1361                                DF55    ;0,0,0,0
1362
1363  002146  006726                ;ITEM 60
1364  002150  015426                EM60    ;INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMR0) SET
1365  002152  020064                DH53    ;TESTNO  ERRORPC
1366  002154  020660                DT53    ;TESTNO,$ERRPC,0
1367                                DF53    ;0,0
1368
1369  002156  007007                ;ITEM 61
1370  002160  015506                EM61    ;NO M.M. TRAP WHEN BIT09 (MMR0) SET PAGE KIPDR7
1371  002162  020104                DH56    ;(MMR0)  KIPDR4  TESTNO  ERRORPC
1372  002164  020666                DT56    ;MMR0,KIPDR4,TESTNO,$ERRPC,0
1373                                DF56    ;0,0,0,0
1374
1375  002166  007066                ;ITEM 62
1376  002170  015446                EM62    ;INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMR0) SET
1377  002172  020072                DH55    ;EXPDATA RECDATA TESTNO  ERRORPC
1378  002174  020662                DT55    ;$REG0,$REG1,TESTNO,$ERRPC,0
1379                                DF55    ;0,0,0,0
1380
1381  002176  007150                ;ITEM 63
1382  002200  015426                EM63    ;TRAP WHEN NO RELOCATION
1383  002202  020064                DH53    ;TESTNO  ERRORPC
1384  002204  020660                DT53    ;TESTNO,$ERRPC,0
1385                                DF53    ;0,0
1386
1387  002206  007200                ;ITEM 64
                                    EM64    ;TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION
```

```
1388
1389  002210  015546                DH64        ;EXPECTING '1074' FOR KERNEL STACK POINTER
1390  002212  020116                DT64        ;RECEIVD TESTNO  ERRORPC
1391  002214  020672                DF64        ;$TMP0,TESTNO,$ERRPC,0
1392                                            ;0,0,0
1393                                ;ITEM 65
1394  002216  007343                EM65        ;MEMORY MANAGEMENT ABORT CONDITION WRONG
1395  002220  015576                DH65        ;EXPCOND ABRTCND TESTNO  ERRORPC
1396  002222  020126                DT65        ;MMEXP,PMMR0,TESTNO,$ERRPC,0
1397  002224  020675                DF65        ;0,0,0,0
1398
1399                                ;ITEM 66
1400  002226  007413                EM66        ;BIT 12 OF MMR0 WAS NOT SET WHEN A.C.F. SATISFIED
1401  002230  015636                DH66        ;(MMR0)  TESTNO  ERRORPC
1402  002232  020140                DT66        ;PMMR0,TESTNO,$ERRPC,0
1403  002234  020701                DF66        ;0,0,0
1404
1405                                ;ITEM 67
1406  002236  007474                EM67        ;NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMR0) CAUSES TRAP COND
1407  002240  015426                DH53        ;TESTNO  ERRORPC
1408  002242  020064                DT53        ;TESTNO,$ERRPC,0
1409  002244  020660                DF53        ;0,0
1410
1411                                ;ITEM 70
1412  002246  007575                EM70        ;ERROR DURING M.M. ABORT IN TRAP SEQUENCE
1413                                            ;EXPECTED:
1414  002250  015666                DH70        ;XXX017  040241  173366  000004
1415                                            ;PROSTAT (MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1416                                            ;RECEIVED:
1417  002252  020150                DT70        ;$REG1,PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
1418  002254  020704                DF70        ;0,0,0,0,0,0
1419
1420                                ;ITEM 71
1421  002256  007660                EM71        ;INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)
1422  002260  015426                DH53        ;TESTNO  ERRORPC
1423  002262  020064                DT53        ;TESTNO,$ERRPC,0
1424  002264  020660                DF53        ;0,0
1425
1426                                ;ITEM 72
1427  002266  007745                EM72        ;ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)
1428  002270  016017                DH72        ;EXPSTAT (MMR0)  TESTNO  ERRORPC
1429  002272  020166                DT72        ;$REG1,PMMR0,TESTNO,$ERRPC,0
1430  002274  020712                DF72        ;0,0,0,0
1431
1432                                ;ITEM 73
1433  002276  010026                EM73        ;AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCK
1434  002300  016057                DH73        ;ORIGINAL DATA          NEW DATA
1435                                            ;(MMR0)  (MMR1)  (MMR2)  (MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1436  002302  020200                DT73        ;PMMR0,PMMR1,PMMR2,$TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
1437  002304  020716                DF73        ;0,0,0,0,0,0,0,0
1438
1439                                ;ITEM 74
1440  002306  010130                EM74        ;DID NOT CHANGE MAPPING TO SUPERVISOR MODE
1441  002310  015426                DH53        ;TESTNO  ERRORPC
1442  002312  020064                DT53        ;TESTNO,$ERRPC,0
1443  002314  020660                DF53        ;0,0
```

```
1444
1445                                    ;ITEM 75
1446  002316  010202                    EM75        ;ABORT CONDITION INCORRECT EXPECTING 100051
1447  002320  016210                    DH75        ;RECEIVD (MMR1)  (MMR2)  TESTNO  ERRORPC
1448  002322  020222                    DT75        ;PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
1449  002324  020726                    DF75        ;0,0,0,0,0
1450
1451                                    ;ITEM 76
1452  002326  010255                    EM76        ;DID NOT CHANGE MAPPING TO USER MODE
1453  002330  015426                    DH53        ;TESTNO  ERRORPC
1454  002332  020064                    DT53        ;TESTNO,$ERRPC,0
1455  002334  020660                    DF53        ;0,0
1456
1457                                    ;ITEM 77
1458  002336  010321                    EM77        ;ABORT CONDITION INCORRECT EXPECTING 100153
1459  002340  016210                    DH75        ;RECEIVD (MMR1)  (MMR2)  TESTNO  ERRORPC
1460  002342  020222                    DT75        ;PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
1461  002344  020726                    DF75        ;0,0,0,0,0
1462
1463                                    ;ITEM 100
1464  002346  010374                    EM100       ;MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1465  002350  016260                    DH100       ;VIRTADR (MMR2)  TESTNO  ERRORPC
1466  002352  020236                    DT100       ;$REG1,PMMR2,TESTNO,$ERRPC,0
1467  002354  020733                    DF100       ;0,0,0,0
1468
1469                                    ;ITEM 101
1470  002356  010445                    EM101       ;MMR0 LOCKED UP THE WRONG PAGE NUMBER
1471  002360  016320                    DH101       ;EXPECTD (MMR0)  TESTNO  ERRORPC
1472  002362  020250                    DT101       ;$REG2,PMMR0,TESTNO,$ERRPC,0
1473  002364  020737                    DF101       ;0,0,0,0
1474
1475                                    ;ITEM 102
1476  002366  010512                    EM102       ;W-BIT NOT SET ON WRITE TO PAGE 4
1477  002370  016360                    DH102       ;KIPDR4  TESTNO  ERRORPC
1478  002372  020262                    DT102       ;$TMP0,TESTNO,$ERRPC,0
1479  002374  020743                    DF102       ;0,0,0
1480
1481                                    ;ITEM 103
1482  002376  010553                    EM103       ;W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1483  002400  016360                    DH102       ;KIPDR4  TESTNO  ERRORPC
1484  002402  020262                    DT102       ;$TMP0,TESTNO,$ERRPC,0
1485  002404  020743                    DF102       ;0,0,0
1486
1487                                    ;ITEM 104
1488  002406  010634                    EM104       ;W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1489  002410  016360                    DH102       ;KIPDR4  TESTNO  ERRORPC
1490  002412  020262                    DT102       ;$TMP0,TESTNO,$ERRPC,0
1491  002414  020743                    DF102       ;0,0,0
1492
1493                                    ;ITEM 105
1494  002416  010703                    EM105       ;A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.=4
1495  002420  016360                    DH102       ;KIPDR4  TESTNO  ERRORPC
1496  002422  020262                    DT102       ;$TMP0,TESTNO,$ERRPC,0
1497  002424  020743                    DF102       ;0,0,0
1498
1499                                    ;ITEM 106
```

```
1500  002426  010760  EM106   ;A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1501  002430  016360  DH102   ;KIPDR4  TESTNO  ERRORPC
1502  002432  020262  DT102   ;$TMP0,TESTNO,$ERRPC,0
1503  002434  020743  DF102   ;0,0,0
1504
1505                  ;ITEM 107
1506  002436  011041  EM107   ;A-BIT DID NOT CLEAR ON WRITE TO KIPAR4
1507  002440  016360  DH102   ;KIPDR4  TESTNO  ERRORPC
1508  002442  020262  DT102   ;$TMP0,TESTNO,$ERRPC,0
1509  002444  020743  DF102   ;0,0,0
1510
1511                  ;ITEM 110
1512  002446  011110  EM110   ;W-BIT DID NOT SET ON WRITE TO I/O PAGE
1513  002450  016410  DH110   ;KIPDR7  TESTNO  ERRORPC
1514  002452  020262  DT102   ;$TMP0,TESTNO,$ERRPC,0
1515  002454  020743  DF102   ;0,0,0
1516
1517                  ;ITEM 111
1518  002456  011157  EM111   ;W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE
1519  002460  016410  DH110   ;KIPDR7  TESTNO  ERRORPC
1520  002462  020262  DT102   ;$TMP0,TESTNO,$ERRPC,0
1521  002464  020743  DF102   ;0,0,0
1522
1523                  ;ITEM 112
1524  002466  011246  EM112   ;DUAL MAPPING BETWEEN PAGES
1525  002470  016440  DH112   ;GDPAGE  BDPAGE  TESTNO  ERRORPC
1526  002472  020272  DT112   ;$REG3,$REG1,TESTNO,$ERRPC,0
1527  002474  020746  DF112   ;0,0,0,0
1528
1529                  ;ITEM 113
1530  002476  011301  EM113   ;NO PAGE HAD BOTH ITS A & W BITS SET
1531  002500  016500  DH113   ;TSTPAGE CONTENT TESTNO  ERRORPC
1532  002502  020304  DT113   ;$REG3,$REG0,TESTNO,$ERRPC,0
1533  002504  020752  DF113   ;0,0,0,0
1534
1535                  ;ITEM 114
1536  002506  011345  EM114   ;DID NOT PICK UP CORRECT STACK POINTER
1537  002510  016540  DH114   ;EXPECTD RECEIVD TESTNO  ERRORPC
1538  002512  020316  DT114   ;$REG2,$REG1,TESTNO,$ERRORPC,0
1539  002514  020756  DF114   ;0,0,0,0
1540
1541                  ;ITEM 115
1542  002516  011413  EM115   ;CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1543  002520  015426  DH53    ;TESTNO  ERRORPC
1544  002522  020064  DT53    ;TESTNO,$ERRPC,0
1545  002524  020660  DF53    ;0,0
1546
1547                  ;ITEM 116
1548  002526  011474  EM116   ;WRONG DATA FETCHED BY MFP INSTRUCTION
1549  002530  016600  DH116   ;EXPECTD RECEIVD TESTNO  ERRORPC
1550  002532  020330  DT116   ;$REG0,$REG1,TESTNO,$ERRPC,0
1551  002534  020762  DF116   ;0,0,0,0
1552
1553                  ;ITEM 117
1554  002536  011542  EM117   ;TRIED TO REFERENCE NON-RESIDENT PAGE
1555  002540  016640  DH117   ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
```

```
1556  002542  020342          DT117                   ;PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
1557  002544  020766          DF117                   ;0,0,0,0,0
1558
1559                          ;ITEM 120
1560  002546  011607          EM120                   ;STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1561  002550  016710          DH120                   ;STKPTR  TESTNO  ERRORPC
1562  002552  020356          DT120                   ;$REG1,TESTNO,$ERRPC,0
1563  002554  020773          DF120                   ;0,0,0
1564
1565                          ;ITEM 121
1566  002556  011664          EM121                   ;INCORRECT STORE BY MTP INSTRUCTION
1567  002560  016740          DH121                   ;GDDATA  STORED  TESTNO  ERRORPC
1568  002562  020330          DT116                   ;$REG0,$REG1,TESTNO,$ERRPC,0
1569  002564  020762          DF116                   ;0,0,0,0
1570
1571                          ;ITEM 122
1572  002566  011727          EM122                   ;ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1573  002570  017000          DH122                   ;(PSW)   TESTNO  ERRORPC EXPECTING XXX340
1574  002572  020366          DT122                   ;$REG0,TESTNO,$ERRPC,0
1575  002574  020776          DF122                   ;0,0,0
1576
1577                          ;ITEM 123
1578  002576  012011          EM123                   ;ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX140
1579  002600  017051          DH123                   ;(PSW)   TESTNO  ERRORPC
1580  002602  020366          DT122                   ;$REG0,TESTNO,$ERRPC,0
1581  002604  020776          DF122                   ;0,0,0
1582
1583                          ;ITEM 124
1584  002606  012101          EM124                   ;ABORTED THRU USER SPACE, USER PSW IS XXX000
1585  002610  017051          DH123                   ;(PSW)   TESTNO  ERRORPC
1586  002612  020366          DT122                   ;$REG0,TESTNO,$ERRPC,0
1587  002614  020776          DF122                   ;0,0,0
1588
1589                          ;ITEM 125
1590  002616  012155          EM125                   ;22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1591  002620  015017          DH40                    ;STARTADR  FINISHADR TESTNO  ERRORPC
1592  002622  017754          DT40                    ;$TMP1,$TMP2,TESTNO,$ERRPC,0
1593  002624  020624          DF40                    ;4,4,0,0
1594
1595                          ;ITEM 126
1596  002626  012236          EM126                   ;22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
1597  002630  015063          DH41                    ;STARTADR  TESTNO  ERRORPC
1598  002632  017766          DT41                    ;$TMP1,TESTNO,$ERRPC,0
1599  002634  020630          DF41                    ;4,0,0
1600
1601                          ;ITEM 127
1602  002636  012320          EM127                   ;DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1603  002640  017101          DH127                   ;TESTNO  ERRORPC
1604  002642  020376          DT127                   ;TESTNO,$ERRPC,0
1605  002644  021001          DF127                   ;0,0
1606
1607                          ;ITEM 130
1608  002646  012410          EM130                   ;ABORT IN KERNEL D-SPACE PICKED UP VECTOR FROM I-SPACE
1609  002650  017000          DH122                   ;(PSW)   TESTNO  ERRORPC EXPECTING XXX340
1610  002652  020366          DT122                   ;$REG0,TESTNO,$ERRPC,0
1611  002654  020776          DF122                   ;0,0,0
```

```
1612
1613                                    ;ITEM 131
1614  002656  012476                    EM131           ;M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION
1615  002660  017120                    DH131           ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC EXPECTING 020031
1616  002662  020404                    DT131           ;PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
1617  002664  021003                    DF131           ;0,0,0,0,0
1618
1619                                    ;ITEM 132
1620  002666  012557                    EM132           ;D SPACE ENABLE CIRCUITRY HAS FAILED
1621  002670  013410                    DH5             ;ERROR   AUTOI/D VIRTUAL
1622                                                    ;REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT
1623  002672  017446                    DT5             ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
1624  002674  020500                    DF5             ;0,0,0,0,0
1625
1626                                    ;ITEM 133
1627  002676  012623                    EM133           ;DPAR READ BACK INCORRECTLY
1628  002700  017211                    DH133           ;ADDR EXP  REC  TEST  ERR
1629  002702  020420                    DT133           ;$TMP0,$TMP1,$TMP2,TESTNO,ERRORPC,0
1630  002704  020602                    DF31            ;0,0,0,0,0
1631
1632  002706                            ER200:          ;STARTING ADDRESS FOR ITEM 201-377
1633
1634                                    ;ITEM 201
1635  002706  012757                    EM201           ;THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1636  002710  017260                    DH201           ;ADDRESS TESTNO
1637
1638                                    ;ITEM 301
1639  002712  020434                    DT201           ;$REG0,,TESTNO,0
1640  002714  021010                    DF201           ;0,0
1641
1642                                    ;ITEM 202
1643  002716  013034                    EM202           ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1644  002720  017277                    DH202           ;ADDRESS ADDRESS
1645                                                    ;LOADED  JUST READ TESTNO
1646
1647                                    ;ITEM 302
1648  002722  020442                    DT202           ;$REG0,$REG1,TESTNO,0
1649  002724  021012                    DF202           ;0,0,0
1650
1651                                    ;ITEM 203
1652  002726  013125                    EM203           ;THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1653  002730  017347                    DH203           ;ADDRESS DATA RED PATTERN COUNT   TESTNO
1654
1655                                    ;ITEM 303
1656  002732  020452                    DT203           ;$REG0,$REG2,$REG4,$REG1,TESTNO,0
1657  002734  021015                    DF203           ;0,0,0,0,0
1658
1659
1660                                    .SBTTL  ERROR TABLE MESSAGES AND DATA POINTERS
1661  002736  047516  020124  044124    EM1:    .ASCIZ  ?NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)?
1662  002744  020105  047503  051122
1663  002752  041505  020124  050103
1664  002760  020125  041101  051117
1665  002766  020124  047503  042116
1666  002774  052111  047511  020116
1667  003002  044124  052522  023440
```

```
 1668  003010  051105  053122  041505
 1669  003016  020047  030050  032060
 1670  003024  000051
 1671  003026  047125  054105  042520  EM2:    .ASCIZ  ?UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?
 1672  003034  052103  042105  041440
 1673  003042  052520  052040  040522
 1674  003050  020120  051117  040440
 1675  003056  047502  052122  052040
 1676  003064  051110  020125  042447
 1677  003072  051122  042526  023503
 1678  003100  024040  030060  024464
 1679  003106     000
 1680  003107     125  042516  050130  EM3:    .ASCII  ?UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
 1681  003114  041505  042524  020104
 1682  003122  040503  044103  020105
 1683  003130  040520  044522  054524
 1684  003136  042440  051122  051117
 1685  003144  052040  051110  020125
 1686  003152  041447  041501  053110
 1687  003160  041505  020047  030450
 1688  003166  032061  100051
 1689  003172  044527  046114  051040          .ASCIZ  ?WILL RETRY THIS TEST ONCE.?
 1690  003200  052105  054522  052040
 1691  003206  044510  020123  042524
 1692  003214  052123  047440  041516
 1693  003222  027105     000
 1694  003225     125  042516  050130  EM4:    .ASCII  ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
 1695  003232  041505  042524  020104
 1696  003240  040515  047111  046440
 1697  003246  046505  051117  020131
 1698  003254  040520  044522  054524
 1699  003262  042440  051122  051117
 1700  003270  052040  051110  020125
 1701  003276  041447  041501  053110
 1702  003304  041505  020047  030450
 1703  003312  032061  100051
 1704  003316  044527  046114  051040          .ASCIZ  ?WILL RETRY THIS TEST ONCE?
 1705  003324  052105  054522  052040
 1706  003332  044510  020123  042524
 1707  003340  052123  047440  041516
 1708  003346  000105
 1709
 1710  003350  047125  054105  042520  EM5:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
 1711  003356  052103  042105  046440
 1712  003364  046505  051117  020131
 1713  003372  040515  040516  042507
 1714  003400  042515  052116  052040
 1715  003406  040522  020120  051117
 1716  003414  040440  047502  052122
 1717  003422     000
 1718  003423     115  046505  051117  EM6:    .ASCIZ  ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
 1719  003430  020131  040515  040516
 1720  003436  042507  042515  052116
 1721  003444  052040  040522  020120
 1722  003452  051117  040440  047502
 1723  003460  052122  044040  042101
```

```
1724  003466  044440  041516  051117
1725  003474  042522  052103  041440
1726  003502  047117  044504  044524
1727  003510  047117     000
1728  003513     115  046505  051117  EM7:    .ASCIZ  ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1729  003520  020131  040515  040516
1730  003526  042507  042515  052116
1731  003534  051040  043505  051511
1732  003542  042524  020122  020060
1733  003550  044527  046114  047040
1734  003556  052117  041440  042514
1735  003564  051101     000
1736  003567     103  047101  052047  EM10:   .ASCIZ  ?CAN'T SET 171000 IN MMR0?
1737  003574  051440  052105  030440
1738  003602  030467  030060  020060
1739  003610  047111  046440  051115
1740  003616  000060
1741  003620  047507  020124  044124  EM11:   .ASCIZ  ?GOT THE WRONG DATA BACK FROM MMR0?
1742  003626  020105  051127  047117
1743  003634  020107  040504  040524
1744  003642  041040  041501  020113
1745  003650  051106  046517  046440
1746  003656  051115  000060
1747  003662  042515  047515  054522  EM12:   .ASCIZ  ?MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR?
1748  003670  046440  047101  043501
1749  003676  046505  047105  020124
1750  003704  042522  044507  052123
1751  003712  051105  030440  053440
1752  003720  046111  020114  047516
1753  003726  020124  046103  040505
1754  003734  000122
1755  003736  046515  030522  042040  EM13:   .ASCIZ  ?MMR1 DID NOT TRACK PROPERLY?
1756  003744  042111  047040  052117
1757  003752  052040  040522  045503
1758  003760  050040  047522  042520
1759  003766  046122  000131
1760  003772  046515  031122  042040  EM14:   .ASCIZ  ?MMR2 DID NOT TRACK PROPERLY?
1761  004000  042111  047040  052117
1762  004006  052040  040522  045503
1763  004014  050040  047522  042520
1764  004022  046122  000131
1765  004026  042515  047515  054522  EM15:   .ASCIZ  ?MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR?
1766  004034  046440  047101  043501
1767  004042  046505  047105  020124
1768  004050  042522  044507  052123
1769  004056  051105  031440  053440
1770  004064  046111  020114  047516
1771  004072  020124  046103  040505
1772  004100  000122
1773  004102  046515  031522  044440  EM16:   .ASCIZ  ?MMR3 IS HOLDING WRONG DATA?
1774  004110  020123  047510  042114
1775  004116  047111  020107  051127
1776  004124  047117  020107  040504
1777  004132  040524     000
1778  004135     123  046525  040515  EM17:   .ASCIZ  ?SUMMARY OF PAR/PDR REFERENCE TIMEOUTS?
1779  004142  054522  047440  020106
```

```
1780  004150  040520  027522  042120
1781  004156  020122  042522  042506
1782  004164  042522  041516  020105
1783  004172  044524  042515  052517
1784  004200  051524     000
1785  004203     106  046117  047514  EM20:   .ASCIZ  ?FOLLOWING PAR/PDR WILL NOT CLEAR?
1786  004210  044527  043516  050040
1787  004216  051101  050057  051104
1788  004224  053440  046111  020114
1789  004232  047516  020124  046103
1790  004240  040505  000122
1791  004244  052523  046515  051101  EM21:   .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS?
1792  004252  020131  043117  042040
1793  004260  040525  020114  042101
1794  004266  051104  051505  044523
1795  004274  043516  042440  051122
1796  004302  051117  000123
1797  004306  052523  046515  051101  EM22:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES?
1798  004314  020131  043117  041440
1799  004322  052517  052116  050040
1800  004330  052101  042524  047122
1801  004336  043040  044501  052514
1802  004344  042522  000123
1803  004350  051105  047522  020122  EM23:   .ASCIZ  ?ERROR IN BYTE ADDRESSING OF PAR/PDR?
1804  004356  047111  041040  052131
1805  004364  020105  042101  051104
1806  004372  051505  044523  043516
1807  004400  047440  020106  040520
1808  004406  027522  042120  000122
1809  004414  047117  020105  043117  EM24:   .ASCIZ  ?ONE OF THE REGISTERS TIMED OUT?
1810  004422  052040  042510  051040
1811  004430  043505  051511  042524
1812  004436  051522  052040  046511
1813  004444  042105  047440  052125
1814  004452     000
1815  004453     114  053517  041040  EM25:   .ASCIZ  ?LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES?
1816  004460  052131  020105  043117
1817  004466  046040  053517  051440
1818  004474  055111  020105  042522
1819  004502  044507  052123  051105
1820  004510  044440  020123  047516
1821  004516  020124  046101  020114
1822  004524  047117  051505     000
1823  004531     103  052517  042114  EM26:   .ASCIZ  ?COULD WRITE ONE OF THE SIZE REGISTERS?
1824  004536  053440  044522  042524
1825  004544  047440  042516  047440
1826  004552  020106  044124  020105
1827  004560  044523  042532  051040
1828  004566  043505  051511  042524
1829  004574  051522     000
1830  004577     110  043511  020110  EM27:   .ASCIZ  ?HIGH SIZE REGISTER IS NOT ZERO?
1831  004604  044523  042532  051040
1832  004612  043505  051511  042524
1833  004620  020122  051511  047040
1834  004626  052117  055040  051105
1835  004634  000117
```

```
1836  004636  050103  020125  051105  EM30:   .ASCIZ  ?CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE?
1837  004644  047522  020122  042522
1838  004652  044507  052123  051105
1839  004660  047040  052117  055040
1840  004666  051105  020117  043101
1841  004674  042524  020122  047514
1842  004702  042101  047111  020107
1843  004710  042516  040507  044524
1844  004716  042526  047440  042516
1845  004724     000
1846  004725     114  053517  051105  EM31:   .ASCIZ  ?LOWER BYTE OF P.S.W. NOT CORRECT?
1847  004732  041040  052131  020105
1848  004740  043117  050040  051456
1849  004746  053456  020056  047516
1850  004754  020124  047503  051122
1851  004762  041505  000124
1852  004766  047514  042101  042105  EM32:   .ASCIZ  ?LOADED MORE THAN LOWER BYTE OF MICRO BREAK REGISTER?
1853  004774  046440  051117  020105
1854  005002  044124  047101  046040
1855  005010  053517  051105  041040
1856  005016  052131  020105  043117
1857  005024  046440  041511  047522
1858  005032  041040  042522  045501
1859  005040  051040  043505  051511
1860  005046  042524  000122
1861  005052  044504  047104  052047  EM33:   .ASCIZ  ?DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER?
1862  005060  046040  040517  020104
1863  005066  051120  043517  040522
1864  005074  020115  047111  042524
1865  005102  051122  050125  020124
1866  005110  042522  052521  051505
1867  005116  020124  042522  044507
1868  005124  052123  051105     000
1869  005131     114  040517  042504  EM34:   .ASCIZ  ?LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER?
1870  005136  020104  047515  042522
1871  005144  052040  040510  020116
1872  005152  050125  042520  020122
1873  005160  054502  042524  047440
1874  005166  020106  052123  041501
1875  005174  020113  044514  044515
1876  005202  020124  042522  044507
1877  005210  052123  051105     000
1878  005215     113  051105  042516  EM35:   .ASCIZ  ?KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP & USP?
1879  005222  020114  052123  041501
1880  005230  020113  047520  047111
1881  005236  042524  020122  047516
1882  005244  020124  030461  030060
1883  005252  040440  052106  051105
1884  005260  046040  040517  044504
1885  005266  043516  051440  050123
1886  005274  023040  052440  050123
1887  005302     000
1888  005303     104  040525  020114  EM36:   .ASCIZ  ?DUAL ADDRESSING BETWEEN PAR/PDR GROUPS?
1889  005310  042101  051104  051505
1890  005316  044523  043516  041040
1891  005324  052105  042527  047105
```

```
1892  005332  050040  051101  050057
1893  005340  051104  043440  047522
1894  005346  050125  000123
1895  005352  040502  020104  042522  EM37:   .ASCIZ  ?BAD RELOCATION, ON STORING DATA 18-BIT MAPPING?
1896  005360  047514  040503  044524
1897  005366  047117  020054  047117
1898  005374  051440  047524  044522
1899  005402  043516  042040  052101
1900  005410  020101  034061  041055
1901  005416  052111  046440  050101
1902  005424  044520  043516     000
1903  005431     061  026470  044502  EM40:   .ASCIZ  ?18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
1904  005436  020124  040515  050120
1905  005444  047111  020107  047520
1906  005452  051523  041111  042514
1907  005460  044040  046117  020105
1908  005466  047111  046440  044501
1909  005474  020116  042515  047515
1910  005502  054522  043040  047522
1911  005510  000115
1912  005512  034061  041055  052111  EM41:   .ASCIZ  ?18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
1913  005520  046440  050101  044520
1914  005526  043516  050040  051517
1915  005534  044523  046102  020105
1916  005542  047510  042514  040440
1917  005550  020124  044124  020105
1918  005556  047524  020120  043117
1919  005564  046440  046505  051117
1920  005572  000131
1921  005574  040506  046125  054524  EM42:   .ASCIZ  ?FAULTY CARRY PROPAGATION 18-BIT MAPPING.?
1922  005602  041440  051101  054522
1923  005610  050040  047522  040520
1924  005616  040507  044524  047117
1925  005624  030440  026470  044502
1926  005632  020124  040515  050120
1927  005640  047111  027107     000
1928  005645     116  020117  051124  EM43:   .ASCIZ  ?NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000?
1929  005652  050101  052040  051110
1930  005660  020125  051105  053122
1931  005666  041505  020054  052101
1932  005674  030440  026470  044502
1933  005702  020124  042101  051104
1934  005710  051505  020123  033067
1935  005716  030060  030060     000
1936  005723     104  042111  023516  EM44:   .ASCIZ  ?DIDN'T GET WRAP AROUND TO ADDRESS ZERO?
1937  005730  020124  042507  020124
1938  005736  051127  050101  040440
1939  005744  047522  047125  020104
1940  005752  047524  040440  042104
1941  005760  042522  051523  055040
1942  005766  051105  000117
1943  005772  047516  052040  040522  EM45:   .ASCIZ  ?NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS?
1944  006000  020120  044124  052522
1945  006006  042440  051122  042526
1946  006014  026103  047440  020116
1947  006022  047516  026516  054105
```

```
1948  006030  051511  040524  052116
1949  006036  040440  042104  042522
1950  006044  051523     000
1951  006047     102  042101  051040  EM46:   .ASCIZ  ?BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING?
1952  006054  046105  041517  052101
1953  006062  047511  026116  041440
1954  006070  051101  054522  050040
1955  006076  047522  040520  040507
1956  006104  044524  047117  031040
1957  006112  026462  044502  020124
1958  006120  040515  050120  047111
1959  006126  000107
1960  006130  044504  020104  047516  EM47:   .ASCIZ  ?DID NOT GET UNIBUS ADDRESS?
1961  006136  020124  042507  020124
1962  006144  047125  041111  051525
1963  006152  040440  042104  042522
1964  006160  051523     000
1965  006163     103  046517  040520  EM50:   .ASCIZ  ?COMPARE CIRCUIT FOR TOP OF MEMORY BAD?
1966  006170  042522  041440  051111
1967  006176  052503  052111  043040
1968  006204  051117  052040  050117
1969  006212  047440  020106  042515
1970  006220  047515  054522  041040
1971  006226  042101     000
1972  006231     120  043501  020105  EM51:   .ASCIZ  ?PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS?
1973  006236  042514  043516  044124
1974  006244  040440  047502  052122
1975  006252  040440  020124  051127
1976  006260  047117  020107  044526
1977  006266  052122  040525  020114
1978  006274  042101  051104  051505
1979  006302  000123
1980  006304  047516  050040  043501  EM52:   .ASCIZ  ?NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT?
1981  006312  020105  042514  043516
1982  006320  044124  040440  047502
1983  006326  052122  020054  044127
1984  006334  047105  041440  047117
1985  006342  044504  044524  047117
1986  006350  041440  051117  042522
1987  006356  052103     000
1988  006361     104  042111  047040  EM53:   .ASCIZ  ?DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5?
1989  006366  052117  040440  047502
1990  006374  052122  047440  020116
1991  006402  047516  026516  042522
1992  006410  044523  042504  052116
1993  006416  050040  043501  020105
1994  006424  044513  042120  032522
1995  006432     000
1996  006433     104  042111  047040  EM54:   .ASCIZ  ?DID NOT ABORT ON READ ONLY PAGE KIPDR4?
1997  006440  052117  040440  047502
1998  006446  052122  047440  020116
1999  006454  042522  042101  047440
2000  006462  046116  020131  040520
2001  006470  042507  045440  050111
2002  006476  051104  000064
2003  006502  047111  047503  051122  EM55:   .ASCIZ  ?INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMR0) CLEAR?
```

```
2004  006510  041505  020124  042522
2005  006516  042101  043040  047522
2006  006524  020115  040520  042507
2007  006532  045440  050111  051104
2008  006540  020064  044502  030124
2009  006546  020071  046450  051115
2010  006554  024460  041440  042514
2011  006562  051101  000
2012  006565  116     020117  027115  EM56:   .ASCIZ  ?NO M.M. TRAP WHEN BIT09 (MMR0) SET PAGE KIPDR4?
2013  006572  027115  052040  040522
2014  006600  020120  044127  047105
2015  006606  041040  052111  034460
2016  006614  024040  046515  030122
2017  006622  020051  042523  020124
2018  006630  040520  042507  045440
2019  006636  050111  051104  000064
2020  006644  047111  047503  051122  EM57:   .ASCIZ  ?INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMR0) SET?
2021  006652  041505  020124  042522
2022  006660  042101  043040  047522
2023  006666  020115  040520  042507
2024  006674  045440  050111  051104
2025  006702  026064  041040  052111
2026  006710  034460  024040  046515
2027  006716  030122  020051  042523
2028  006724  000124
2029  006726  047111  047503  051122  EM60:   .ASCIZ  ?INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMR0) SET?
2030  006734  041505  020124  051127
2031  006742  052111  020105  047524
2032  006750  050040  043501  020105
2033  006756  044513  042120  032122
2034  006764  020054  044502  030124
2035  006772  020071  046450  051115
2036  007000  024460  051440  052105
2037  007006  000
2038  007007  116     020117  027115  EM61:   .ASCIZ  ?NO M.M. TRAP WHEN BIT09 (MMR0) SET PAGE KIPDR7?
2039  007014  027115  052040  040522
2040  007022  020120  044127  047105
2041  007030  041040  052111  034460
2042  007036  024040  046515  030122
2043  007044  020051  042523  020124
2044  007052  040520  042507  045440
2045  007060  050111  051104  000067
2046  007066  047111  047503  051122  EM62:   .ASCIZ  ?INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMR0) SET?
2047  007074  041505  020124  042522
2048  007102  042101  043040  047522
2049  007110  020115  040520  042507
2050  007116  045440  050111  051104
2051  007124  026067  041040  052111
2052  007132  034460  024040  046515
2053  007140  030122  020051  042523
2054  007146  000124
2055  007150  051124  050101  053440  EM63:   .ASCIZ  ?TRAP WHEN NO RELOCATION?
2056  007156  042510  020116  047516
2057  007164  051040  046105  041517
2058  007172  052101  047511  000116
2059  007200  047524  045517  052040  EM64:   .ASCII  ?TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION?<CRLF>
```

```
 2060  007206  047527  053040  041505
 2061  007214  047524  051522  053440
 2062  007222  052111  020110  051124
 2063  007230  050101  040440  042116
 2064  007236  040440  047502  052122
 2065  007244  047440  020116  040523
 2066  007252  042515  044440  051516
 2067  007260  051124  041525  044524
 2068  007266  047117     200
 2069  007271     105  050130  041505          .ASCIZ  ?EXPECTING ''1074'' FOR KERNEL STACK POINTER?
 2070  007276  044524  043516  021040
 2071  007304  030061  032067  020042
 2072  007312  047506  020122  042513
 2073  007320  047122  046105  051440
 2074  007326  040524  045503  050040
 2075  007334  044517  052116  051105
 2076  007342     000
 2077  007343     115  046505  051117  EM65:   .ASCIZ  ?MEMORY MANAGEMENT ABORT CONDITION WRONG?
 2078  007350  020131  040515  040516
 2079  007356  042507  042515  052116
 2080  007364  040440  047502  052122
 2081  007372  041440  047117  044504
 2082  007400  044524  047117  053440
 2083  007406  047522  043516     000
 2084  007413     102  052111  030440  EM66:   .ASCIZ  ?BIT 12 OF MMR0 WAS NOT SET WHEN A.C.F. SATISFIED?
 2085  007420  020062  043117  046440
 2086  007426  051115  020060  040527
 2087  007434  020123  047516  020124
 2088  007442  042523  020124  044127
 2089  007450  047105  040440  041456
 2090  007456  043056  020056  040523
 2091  007464  044524  043123  042511
 2092  007472  000104
 2093  007474  047516  052040  040522  EM67:   .ASCIZ  ?NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMR0) CAUSES TRAP COND.?
 2094  007502  020120  044127  047105
 2095  007510  044440  051516  051124
 2096  007516  041525  044524  047117
 2097  007524  041440  042514  051101
 2098  007532  047111  020107  044502
 2099  007540  030124  020071  046450
 2100  007546  051115  024460  041440
 2101  007554  052501  042523  020123
 2102  007562  051124  050101  041440
 2103  007570  047117  027104     000
 2104  007575     105  051122  051117  EM70:   .ASCII  ?ERROR DURING M.M. ABORT IN TRAP SEQUENCE?<CRLF>
 2105  007602  042040  051125  047111
 2106  007610  020107  027115  027115
 2107  007616  040440  047502  052122
 2108  007624  044440  020116  051124
 2109  007632  050101  051440  050505
 2110  007640  042525  041516  100105
 2111  007646  054105  042520  052103          .ASCIZ  ?EXPECTED:?
 2112  007654  042105  000072
 2113  007660  047111  052123  052522  EM71:   .ASCIZ  ?INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)?
 2114  007666  052103  047511  020116
 2115  007674  042506  041524  020110
```

```
2116  007702  044504  020104  047516
2117  007710  020124  041101  051117
2118  007716  020124  047111  044440
2119  007724  046114  043505  046101
2120  007732  046440  042117  020105
2121  007740  030450  024460     000
2122  007745     105  051122  051117  EM72:   .ASCIZ  ?ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)?
2123  007752  041440  047117  044504
2124  007760  044524  047117  047040
2125  007766  052117  041440  051117
2126  007774  042522  052103  044440
2127  010002  020116  046111  042514
2128  010010  040507  020114  047515
2129  010016  042504  024040  030061
2130  010024  000051
2131  010026  052101  046040  040505  EM73:   .ASCIZ  ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCKED?
2132  010034  052123  047440  042516
2133  010042  046440  046456  020056
2134  010050  052123  052101  051525
2135  010056  051040  043505  051511
2136  010064  042524  051522  053440
2137  010072  051501  041440  047514
2138  010100  045503  042105  040440
2139  010106  052106  051105  041040
2140  010114  044505  043516  046040
2141  010122  041517  042513  000104
2142  010130  044504  020104  047516  EM74:   .ASCIZ  ?DID NOT CHANGE MAPPING TO SUPERVISOR MODE?
2143  010136  020124  044103  047101
2144  010144  042507  046440  050101
2145  010152  044520  043516  052040
2146  010160  020117  052523  042520
2147  010166  053122  051511  051117
2148  010174  046440  042117  000105
2149  010202  041101  051117  020124  EM75:   .ASCIZ  ?ABORT CONDITION INCORRECT EXPECTING 100051?
2150  010210  047503  042116  052111
2151  010216  047511  020116  047111
2152  010224  047503  051122  041505
2153  010232  020124  054105  042520
2154  010240  052103  047111  020107
2155  010246  030061  030060  030465
2156  010254     000
2157  010255     104  042111  047040  EM76:   .ASCIZ  ?DID NOT CHANGE MAPPING TO USER MODE?
2158  010262  052117  041440  040510
2159  010270  043516  020105  040515
2160  010276  050120  047111  020107
2161  010304  047524  052440  042523
2162  010312  020122  047515  042504
2163  010320     000
2164  010321     101  047502  052122  EM77:   .ASCIZ  ?ABORT CONDITION INCORRECT EXPECTING 100153?
2165  010326  041440  047117  044504
2166  010334  044524  047117  044440
2167  010342  041516  051117  042522
2168  010350  052103  042440  050130
2169  010356  041505  044524  043516
2170  010364  030440  030060  032461
2171  010372  000063
```

```
 2172  010374  046515  031122  046040  EM100:  .ASCIZ  ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?
 2173  010402  041517  042513  020104
 2174  010410  050125  052040  042510
 2175  010416  053440  047522  043516
 2176  010424  053040  051111  052524
 2177  010432  046101  040440  042104
 2178  010440  042522  051523     000
 2179  010445     115  051115  020060  EM101:  .ASCIZ  ?MMR0 LOCKED UP THE WRONG PAGE NUMBER?
 2180  010452  047514  045503  042105
 2181  010460  052440  020120  044124
 2182  010466  020105  051127  047117
 2183  010474  020107  040520  042507
 2184  010502  047040  046525  042502
 2185  010510  000122
 2186  010512  026527  044502  020124  EM102:  .ASCIZ  ?W-BIT NOT SET ON WRITE TO PAGE 4?
 2187  010520  047516  020124  042523
 2188  010526  020124  047117  053440
 2189  010534  044522  042524  052040
 2190  010542  020117  040520  042507
 2191  010550  032040     000
 2192  010553     127  041055  052111  EM103:  .ASCIZ  ?W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
 2193  010560  042040  042111  047040
 2194  010566  052117  051040  046505
 2195  010574  044501  020116  042523
 2196  010602  020124  047117  046440
 2197  010610  046456  020056  041101
 2198  010616  051117  020124  052101
 2199  010624  050040  043501  020105
 2200  010632  000064
 2201  010634  026527  044502  020124  EM104:  .ASCIZ  ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
 2202  010642  044504  020104  047516
 2203  010650  020124  046103  040505
 2204  010656  020122  047117  053440
 2205  010664  044522  042524  052040
 2206  010672  020117  044513  042120
 2207  010700  032122     000
 2208  010703     101  041055  052111  EM105:  .ASCIZ  ?A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.=4?
 2209  010710  042040  042111  047040
 2210  010716  052117  051440  052105
 2211  010724  047440  020116  042522
 2212  010732  042101  052040  020117
 2213  010740  040520  042507  032040
 2214  010746  040440  041456  043056
 2215  010754  036456  000064
 2216  010760  026501  044502  020124  EM106:  .ASCIZ  ?A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
 2217  010766  044504  020104  047516
 2218  010774  020124  042522  040515
 2219  011002  047111  051440  052105
 2220  011010  047440  020116  027115
 2221  011016  027115  040440  047502
 2222  011024  052122  040440  020124
 2223  011032  040520  042507  032040
 2224  011040     000
 2225  011041     101  041055  052111  EM107:  .ASCIZ  ?A-BIT DID NOT CLEAR ON WRITE TO KIPAR4?
 2226  011046  042040  042111  047040
 2227  011054  052117  041440  042514
```

```
 2228  011062  051101  047440  020116
 2229  011070  051127  052111  020105
 2230  011076  047524  045440  050111
 2231  011104  051101  000064
 2232  011110  026527  044502  020124  EM110:  .ASCIZ  ?W-BIT DID NOT SET ON WRITE TO I/O PAGE?
 2233  011116  044504  020104  047516
 2234  011124  020124  042523  020124
 2235  011132  047117  053440  044522
 2236  011140  042524  052040  020117
 2237  011146  027511  020117  040520
 2238  011154  042507     000
 2239  011157     127  041055  052111  EM111:  .ASCIZ  ?W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE?
 2240  011164  042040  042111  047040
 2241  011172  052117  051040  046505
 2242  011200  044501  020116  042523
 2243  011206  020124  043101  042524
 2244  011214  020122  047111  042524
 2245  011222  047122  046101  051040
 2246  011230  043505  051511  042524
 2247  011236  020122  051127  052111
 2248  011244  000105
 2249  011246  052504  046101  046440  EM112:  .ASCIZ  ?DUAL MAPPING BETWEEN PAGES?
 2250  011254  050101  044520  043516
 2251  011262  041040  052105  042527
 2252  011270  047105  050040  043501
 2253  011276  051505     000
 2254  011301     116  020117  040520  EM113:  .ASCIZ  ?NO PAGE HAD BOTH ITS A & W BITS SET?
 2255  011306  042507  044040  042101
 2256  011314  041040  052117  020110
 2257  011322  052111  020123  020101
 2258  011330  020046  020127  044502
 2259  011336  051524  051440  052105
 2260  011344     000
 2261  011345     104  042111  047040  EM114:  .ASCIZ  ?DID NOT PICK UP CORRECT STACK POINTER?
 2262  011352  052117  050040  041511
 2263  011360  020113  050125  041440
 2264  011366  051117  042522  052103
 2265  011374  051440  040524  045503
 2266  011402  050040  044517  052116
 2267  011410  051105     000
 2268  011413     103  051125  042522  EM115:  .ASCIZ  ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
 2269  011420  052116  046440  042117
 2270  011426  020105  052123  041501
 2271  011434  020113  047516  020124
 2272  011442  052520  044123  042105
 2273  011450  044440  020116  043115
 2274  011456  020120  047111  052123
 2275  011464  052522  052103  047511
 2276  011472  000116
 2277  011474  051127  047117  020107  EM116:  .ASCIZ  ?WRONG DATA FETCHED BY MFP INSTRUCTION?
 2278  011502  040504  040524  043040
 2279  011510  052105  044103  042105
 2280  011516  041040  020131  043115
 2281  011524  020120  047111  052123
 2282  011532  052522  052103  047511
 2283  011540  000116
```

```
 2284  011542  051124  042511  020104  EM117:  .ASCIZ  ?TRIED TO REFERENCE NON-RESIDENT PAGE?
 2285  011550  047524  051040  043105
 2286  011556  051105  047105  042503
 2287  011564  047040  047117  051055
 2288  011572  051505  042111  047105
 2289  011600  020124  040520  042507
 2290  011606     000
 2291  011607     123  040524  045503  EM120:  .ASCIZ  ?STACK POINTER NOT CHANGED BY MTP INSTRUCTION?
 2292  011614  050040  044517  052116
 2293  011622  051105  047040  052117
 2294  011630  041440  040510  043516
 2295  011636  042105  041040  020131
 2296  011644  052115  020120  047111
 2297  011652  052123  052522  052103
 2298  011660  047511  000116
 2299  011664  047111  047503  051122  EM121:  .ASCIZ  ?INCORRECT STORE BY MTP INSTRUCTION?
 2300  011672  041505  020124  052123
 2301  011700  051117  020105  054502
 2302  011706  046440  050124  044440
 2303  011714  051516  051124  041525
 2304  011722  044524  047117     000
 2305  011727     101  047502  052122  EM122:  .ASCIZ  ?ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW?
 2306  011734  042105  052040  051110
 2307  011742  020125  044522  044107
 2308  011750  020124  042526  052103
 2309  011756  051117  041040  052125
 2310  011764  050040  041511  042513
 2311  011772  020104  050125  053440
 2312  012000  047522  043516  050040
 2313  012006  053523     000
 2314  012011     101  047502  052122  EM123:  .ASCIZ  ?ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX340?
 2315  012016  042105  052040  051110
 2316  012024  020125  052523  042520
 2317  012032  053122  051511  051117
 2318  012040  051440  040520  042503
 2319  012046  020054  052523  042520
 2320  012054  053122  051511  051117
 2321  012062  050040  053523  044440
 2322  012070  020123  054130  031530
 2323  012076  030064     000
 2324  012101     101  047502  052122  EM124:  .ASCIZ  ?ABORTED THRU USER SPACE, USER PSW IS XXX000?
 2325  012106  042105  052040  051110
 2326  012114  020125  051525  051105
 2327  012122  051440  040520  042503
 2328  012130  020054  051525  051105
 2329  012136  050040  053523  044440
 2330  012144  020123  054130  030130
 2331  012152  030060     000
 2332  012155     062  026462  044502  EM125:  .ASCIZ  ?22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
 2333  012162  020124  040515  050120
 2334  012170  047111  020107  047520
 2335  012176  051523  041111  042514
 2336  012204  044040  046117  020105
 2337  012212  047111  046440  044501
 2338  012220  020116  042515  047515
 2339  012226  054522  043040  047522
```

```
 2340  012234  000115
 2341  012236  031062  041055  052111  EM126:  .ASCIZ  ?22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
 2342  012244  046440  050101  044520
 2343  012252  043516  050040  051517
 2344  012260  044523  046102  020105
 2345  012266  047510  042514  040440
 2346  012274  020124  044124  020105
 2347  012302  047524  020120  043117
 2348  012310  046440  046505  051117
 2349  012316  000131
 2350  012320  044504  020104  047516  EM127:  .ASCIZ  ?DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER?
 2351  012326  020124  041101  051117
 2352  012334  020124  042522  042506
 2353  012342  042522  041516  020105
 2354  012350  047524  040440  046440
 2355  012356  046505  051117  020131
 2356  012364  040515  040516  042507
 2357  012372  042515  052116  051040
 2358  012400  043505  051511  042524
 2359  012406  000122
 2360  012410  041101  051117  020124  EM130:  .ASCIZ  ?ABORT IN KERNEL D-SPACE PICKED UP VECTOR FROM I-SPACE?
 2361  012416  047111  045440  051105
 2362  012424  042516  020114  026504
 2363  012432  050123  041501  020105
 2364  012440  044520  045503  042105
 2365  012446  052440  020120  042526
 2366  012454  052103  051117  043040
 2367  012462  047522  020115  026511
 2368  012470  050123  041501  000105
 2369  012476  027115  027115  040440  EM131:  .ASCIZ  ?M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION?
 2370  012504  047502  052122  044440
 2371  012512  020116  042513  047122
 2372  012520  046105  042040  051455
 2373  012526  040520  042503  044040
 2374  012534  042101  053440  047522
 2375  012542  043516  041440  047117
 2376  012550  044504  044524  047117
 2377  012556     000
 2378  012557     104  051455  040520  EM132:  .ASCIZ  ?D-SPACE ENABLE CIRCUITRY HAS FAILED?
 2379  012564  042503  042440  040516
 2380  012572  046102  020105  044503
 2381  012600  041522  044525  051124
 2382  012606  020131  040510  020123
 2383  012614  040506  046111  042105
 2384  012622     000
 2385  012623     104  040520  020122  EM133:  .ASCIZ  ?DPAR READ BACK INCORRECTLY?
 2386  012630  042522  042101  041040
 2387  012636  041501  020113  047111
 2388  012644  047503  051122  041505
 2389  012652  046124  000131
 2390  012656  006412  050104  051101  ECO:    .ASCIZ  <12><15>/DPAR READ BACK ERROR MAY BE CORRECTED BY ECO NO. M8140-00002/<1
 2391  012664  051040  040505  020104
 2392  012672  040502  045503  042440
 2393  012700  051122  051117  046440
 2394  012706  054501  041040  020105
 2395  012714  047503  051122  041505
```

```
 2396  012722  042524  020104  054502
 2397  012730  042440  047503  047040
 2398  012736  027117  046440  030470
 2399  012744  030064  030055  030060
 2400  012752  031060  006412     000
 2401
 2402  012757     124  042510  043040  EM201:  .ASCIZ  ?THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS?
 2403  012764  046117  047514  044527
 2404  012772  043516  040440  042522
 2405  013000  050040  051101  050057
 2406  013006  051104  051040  043105
 2407  013014  051105  047105  042503
 2408  013022  052040  046511  047505
 2409  013030  052125  000123
 2410  013034  044124  020105  047506  EM202:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S?
 2411  013042  046114  053517  047111
 2412  013050  020107  051101  020105
 2413  013056  052504  046101  040440
 2414  013064  042104  042522  051523
 2415  013072  047111  020107  051105
 2416  013100  047522  051522  043040
 2417  013106  051117  050040  051101
 2418  013114  051447  050057  051104
 2419  013122  051447     000
 2420  013125     124  042510  043040  EM203:  .ASCIZ  ?THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S?
 2421  013132  046117  047514  044527
 2422  013140  043516  040440  042522
 2423  013146  041440  052517  052116
 2424  013154  050040  052101  042524
 2425  013162  047122  042440  051122
 2426  013170  051117  020123  047506
 2427  013176  020122  040520  023522
 2428  013204  027523  042120  023522
 2429  013212  000123
 2430
 2431
 2432  013214  054105  042520  052103  DH1:    .ASCII  ?EXPECTD ?
 2433  013222  020104
 2434  013224  042522  042503  053111  DH2:    .ASCIZ  ?RECEIVD TESTNO  PC AT ABORT?
 2435  013232  020104  042524  052123
 2436  013240  047516  020040  041520
 2437  013246  040440  020124  041101
 2438  013254  051117  000124
 2439  013260  040520  044522  054524  DH3:    .ASCII  ?PARITY  ADDRESS   CONTROL MAINTEN?<CRLF>
 2440  013266  020040  042101  051104
 2441  013274  051505  020123  020040
 2442  013302  047503  052116  047522
 2443  013310  020114  040515  047111
 2444  013316  042524  100116
 2445  013322  047503  042116  052111          .ASCIZ  ?CONDITN REFERENCD REGISTR REGISTR TESTNO  PC AT ABORT?
 2446  013330  020116  042522  042506
 2447  013336  042522  041516  020104
 2448  013344  042522  044507  052123
 2449  013352  020122  042522  044507
 2450  013360  052123  020122  042524
 2451  013366  052123  047516  020040
```

```
 2452  013374  041520  040440  020124
 2453  013402  041101  051117  000124
 2454  013410  051105  047522  020122  DH5:    .ASCII  ?ERROR   AUTOI/D VIRTUAL?<CRLF>
 2455  013416  020040  052501  047524
 2456  013424  027511  020104  044526
 2457  013432  052122  040525  100114
 2458  013440  042522  044507  052123          .ASCIZ  ?REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT?
 2459  013446  020122  042522  044507
 2460  013454  052123  020122  042101
 2461  013462  051104  051505  020123
 2462  013470  042524  052123  047516
 2463  013476  020040  041520  040440
 2464  013504  020124  041101  051117
 2465  013512  000124
 2466  013514  054105  042520  052103  DH6:    .ASCII  ?EXPECTD ERROR   AUTOI/D VIRTUAL?<CRLF>
 2467  013522  020104  051105  047522
 2468  013530  020122  020040  052501
 2469  013536  047524  027511  020104
 2470  013544  044526  052122  040525
 2471  013552  100114
 2472  013554  047503  042116  052111          .ASCIZ  ?CONDITN REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT?
 2473  013562  020116  042522  044507
 2474  013570  052123  020122  042522
 2475  013576  044507  052123  020122
 2476  013604  042101  051104  051505
 2477  013612  020123  042524  052123
 2478  013620  047516  020040  041520
 2479  013626  040440  020124  041101
 2480  013634  051117  000124
 2481  013640  046450  051115  024460  DH7:    .ASCIZ  ?(MMR0)  TESTNO  ERRORPC?
 2482  013646  020040  042524  052123
 2483  013654  047516  020040  051105
 2484  013662  047522  050122  000103
 2485  013670  047514  042101  042105  DH11:   .ASCIZ  ?LOADED  RECEIVD TESTNO  ERRORPC?
 2486  013676  020040  042522  042503
 2487  013704  053111  020104  042524
 2488  013712  052123  047516  020040
 2489  013720  051105  047522  050122
 2490  013726  000103
 2491  013730  054105  042520  052103  DH13:   .ASCII  ?EXPECTD ?
 2492  013736  020104
 2493  013740  046450  051115  024461  DH12:   .ASCIZ  ?(MMR1)  TESTNO  ERRORPC?
 2494  013746  020040  042524  052123
 2495  013754  047516  020040  051105
 2496  013762  047522  050122  000103
 2497  013770  054105  042520  052103  DH14:   .ASCIZ  ?EXPECTD (MMR2)  TESTNO  ERRORPC?
 2498  013776  020104  046450  051115
 2499  014004  024462  020040  042524
 2500  014012  052123  047516  020040
 2501  014020  051105  047522  050122
 2502  014026  000103
 2503  014030  047514  042101  042105  DH16:   .ASCII  ?LOADED  ?
 2504  014036  020040
 2505  014040  046450  051115  024463  DH15:   .ASCIZ  ?(MMR3)  TESTNO  ERRORPC?
 2506  014046  020040  042524  052123
 2507  014054  047516  020040  051105
```

```
 2508  014062  047522  050122  000103
 2509  014070  042101  051104  051117  DH17:   .ASCIZ  ?ADDROR  ADDRAND TESTNO  #ERRORS?
 2510  014076  020040  042101  051104
 2511  014104  047101  020104  042524
 2512  014112  052123  047516  020040
 2513  014120  042443  051122  051117
 2514  014126  000123
 2515  014130  042101  051104  051505  DH20:   .ASCIZ  ?ADDRESS DATA    TESTNO  ERRORPC?
 2516  014136  020123  040504  040524
 2517  014144  020040  020040  042524
 2518  014152  052123  047516  020040
 2519  014160  051105  047522  050122
 2520  014166  000103
 2521  014170  042101  051104  051117  DH21:   .ASCII  ?ADDROR  ADDRAND ADDROR  ADDRAND?<CRLF>
 2522  014176  020040  042101  051104
 2523  014204  047101  020104  042101
 2524  014212  051104  051117  020040
 2525  014220  042101  051104  047101
 2526  014226  100104
 2527  014230  047514  042101  042105          .ASCIZ  ?LOADED  LOADED  ENABLED ENABLED TESTNO  #ERRORS?
 2528  014236  020040  047514  042101
 2529  014244  042105  020040  047105
 2530  014252  041101  042514  020104
 2531  014260  047105  041101  042514
 2532  014266  020104  042524  052123
 2533  014274  047516  020040  042443
 2534  014302  051122  051117  000123
 2535  014310  042101  051104  051117  DH22:   .ASCIZ  ?ADDROR  ADDRAND PATRNOR PATRNAD DATAOR  DATAND TESTNO  #ERRORS?
 2536  014316  020040  042101  051104
 2537  014324  047101  020104  040520
 2538  014332  051124  047516  020122
 2539  014340  040520  051124  040516
 2540  014346  020104  040504  040524
 2541  014354  051117  020040  040504
 2542  014362  040524  042116  052040
 2543  014370  051505  047124  020117
 2544  014376  021440  051105  047522
 2545  014404  051522     000
 2546  014407     101  042104  042522  DH23:   .ASCIZ  ?ADDRESS EXPECTD RECEIVD TESTNO  ERRORPC?
 2547  014414  051523  042440  050130
 2548  014422  041505  042124  051040
 2549  014430  041505  044505  042126
 2550  014436  052040  051505  047124
 2551  014444  020117  042440  051122
 2552  014452  051117  041520     000
 2553  014457     122  043505  042101  DH24:   .ASCIZ  ?REGADDR TESTNO  ERRORPC?
 2554  014464  051104  052040  051505
 2555  014472  047124  020117  042440
 2556  014500  051122  051117  041520
 2557  014506     000
 2558  014507     122  043505  042101  DH25:   .ASCIZ  ?REGADDR DATA    TESTNO  ERRORPC?
 2559  014514  051104  042040  052101
 2560  014522  020101  020040  052040
 2561  014530  051505  047124  020117
 2562  014536  042440  051122  051117
 2563  014544  041520     000
```

```
 2564  014547     122  043505  042101  DH31:   .ASCIZ  ?REGADDR DATAREC DATAEXP TESTNO  ERRORPC?
 2565  014554  051104  042040  052101
 2566  014562  051101  041505  042040
 2567  014570  052101  042501  050130
 2568  014576  052040  051505  047124
 2569  014604  020117  042440  051122
 2570  014612  051117  041520     000
 2571  014617     113  050123  020040  DH35:   .ASCIZ  ?KSP     TESTNO  ERRORPC?
 2572  014624  020040  052040  051505
 2573  014632  047124  020117  042440
 2574  014640  051122  051117  041520
 2575  014646     000
 2576  014647     111  042116  054105  DH36:   .ASCII  ?INDEX   INDEX   PAR/PDR?<CRLF>
 2577  014654  020040  044440  042116
 2578  014662  054105  020040  050040
 2579  014670  051101  050057  051104
 2580  014676     200
 2581  014677     105  050130  041505          .ASCIZ  ?EXPECTD RECEIVD ADRREAD TESTNO  ERRORPC?
 2582  014704  042124  051040  041505
 2583  014712  044505  042126  040440
 2584  014720  051104  042522  042101
 2585  014726  052040  051505  047124
 2586  014734  020117  042440  051122
 2587  014742  051117  041520     000
 2588  014747     101  042104  042522  DH37:   .ASCIZ  ?ADDRESS GDDATA  BADDATA TESTNO  ERRORPC?
 2589  014754  051523  043440  042104
 2590  014762  052101  020101  041040
 2591  014770  042101  040504  040524
 2592  014776  052040  051505  047124
 2593  015004  020117  042440  051122
 2594  015012  051117  041520     000
 2595  015017     123  040524  052122  DH40:   .ASCIZ  ?STARTADR  FINISHADR TESTNO  ERRORPC?
 2596  015024  042101  020122  043040
 2597  015032  047111  051511  040510
 2598  015040  051104  052040  051505
 2599  015046  047124  020117  042440
 2600  015054  051122  051117  041520
 2601  015062     000
 2602  015063     123  040524  052122  DH41:   .ASCIZ  ?STARTADR  TESTNO  ERRORPC?
 2603  015070  042101  020122  052040
 2604  015076  051505  047124  020117
 2605  015104  042440  051122  051117
 2606  015112  041520     000
 2607  015115     120  052101  042524  DH42:   .ASCII  ?PATTERN DATA    ADDRESS?<CRLF>
 2608  015122  047122  042040  052101
 2609  015130  020101  020040  040440
 2610  015136  042104  042522  051523
 2611  015144     200
 2612  015145     114  040517  042504          .ASCIZ  ?LOADED  FETCHED INTENDED  TESTNO  ERRORPC?
 2613  015152  020104  043040  052105
 2614  015160  044103  042105  044440
 2615  015166  052116  047105  042504
 2616  015174  020104  052040  051505
 2617  015202  047124  020117  042440
 2618  015210  051122  051117  041520
 2619  015216     000
```

```
2620  015217     124  051505  047124  DH43:   .ASCIZ  ?TESTNO  ERRORPC?
2621  015224  020117  042440  051122
2622  015232  051117  041520     000
2623  015237     104  052101  020101  DH44:   .ASCIZ  ?DATA    TESTNO  ERRORPC?
2624  015244  020040  052040  051505
2625  015252  047124  020117  042440
2626  015260  051122  051117  041520
2627  015266     000
2628  015267     116  047117  042455  DH45:   .ASCIZ  ?NON-EXADDR TESTNO  ERRORPC?
2629  015274  040530  042104  020122
2630  015302  042524  052123  047516
2631  015310  020040  051105  047522
2632  015316  050122  000103
2633  015322  044513  040520  032122  DH50:   .ASCIZ  ?KIPAR4    SIZELO    TESTNO  ERRORPC?
2634  015330  020040  020040  044523
2635  015336  042532  047514  020040
2636  015344  020040  042524  052123
2637  015352  047516  020040  051105
2638  015360  047522  050122  000103
2639  015366  043520  042514  043116  DH51:   .ASCIZ  ?PGLENFD VABLKNO TESTNO  ERRORPC?
2640  015374  020104  040526  046102
2641  015402  047113  020117  042524
2642  015410  052123  047516  020040
2643  015416  051105  047522  050122
2644  015424  000103
2645  015426  042524  052123  047516  DH53:   .ASCIZ  ?TESTNO  ERRORPC?
2646  015434  020040  051105  047522
2647  015442  050122  000103
2648  015446  054105  042120  052101  DH55:   .ASCIZ  ?EXPDATA RECDATA TESTNO  ERRORPC?
2649  015454  020101  042522  042103
2650  015462  052101  020101  042524
2651  015470  052123  047516  020040
2652  015476  051105  047522  050122
2653  015504  000103
2654  015506  046450  051115  024460  DH56:   .ASCIZ  ?(MMR0)  KIPDR4  TESTNO  ERRORPC?
2655  015514  020040  044513  042120
2656  015522  032122  020040  042524
2657  015530  052123  047516  020040
2658  015536  051105  047522  050122
2659  015544  000103
2660  015546  042522  042503  053111  DH64:   .ASCIZ  ?RECEIVD TESTNO  ERRORPC?
2661  015554  020104  042524  052123
2662  015562  047516  020040  051105
2663  015570  047522  050122  000103
2664  015576  054105  041520  047117  DH65:   .ASCIZ  ?EXPCOND ABRTCND TESTNO  ERRORPC?
2665  015604  020104  041101  052122
2666  015612  047103  020104  042524
2667  015620  052123  047516  020040
2668  015626  051105  047522  050122
2669  015634  000103
2670  015636  046450  051115  024460  DH66:   .ASCIZ  ?(MMR0)  TESTNO  ERRORPC?
2671  015644  020040  042524  052123
2672  015652  047516  020040  051105
2673  015660  047522  050122  000103
2674  015666  054130  030130  033461  DH70:   .ASCII  ?XXX017  040241  173366  000004?<CRLF>
2675  015674  020040  032060  031060
```

```
 2676  015702  030464  020040  033461
 2677  015710  031463  033066  020040
 2678  015716  030060  030060  032060
 2679  015724     200
 2680  015725     120  047522  052123          .ASCII  ?PROSTAT (MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC?<CRLF>
 2681  015732  052101  024040  046515
 2682  015740  030122  020051  024040
 2683  015746  046515  030522  020051
 2684  015754  024040  046515  031122
 2685  015762  020051  052040  051505
 2686  015770  047124  020117  042440
 2687  015776  051122  051117  041520
 2688  016004     200
 2689  016005     122  041505  044505          .ASCIZ  ?RECEIVED:?
 2690  016012  042526  035104     000
 2691  016017     105  050130  052123  DH72:   .ASCIZ  ?EXPSTAT (MMR0)  TESTNO  ERRORPC?
 2692  016024  052101  024040  046515
 2693  016032  030122  020051  052040
 2694  016040  051505  047124  020117
 2695  016046  042440  051122  051117
 2696  016054  041520     000
 2697  016057     117  044522  044507  DH73:   .ASCII  ?ORIGINAL DATA           NEW DATA?<CRLF>
 2698  016064  040516  020114  040504
 2699  016072  040524  004411  047040
 2700  016100  053505  042040  052101
 2701  016106  100101
 2702  016110  046450  051115  024460          .ASCIZ  ?(MMR0)  (MMR1)  (MMR2)  (MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC?
 2703  016116  020040  046450  051115
 2704  016124  024461  020040  046450
 2705  016132  051115  024462  020040
 2706  016140  046450  051115  024460
 2707  016146  020040  046450  051115
 2708  016154  024461  020040  046450
 2709  016162  051115  024462  020040
 2710  016170  042524  052123  047516
 2711  016176  020040  051105  047522
 2712  016204  050122  000103
 2713  016210  042522  042503  053111  DH75:   .ASCIZ  ?RECEIVD (MMR1)  (MMR2)  TESTNO  ERRORPC?
 2714  016216  020104  046450  051115
 2715  016224  024461  020040  046450
 2716  016232  051115  024462  020040
 2717  016240  042524  052123  047516
 2718  016246  020040  051105  047522
 2719  016254  050122  000103
 2720  016260  044526  052122  042101  DH100:  .ASCIZ  ?VIRTADR (MMR2)  TESTNO  ERRORPC?
 2721  016266  020122  046450  051115
 2722  016274  024462  020040  042524
 2723  016302  052123  047516  020040
 2724  016310  051105  047522  050122
 2725  016316  000103
 2726  016320  054105  042520  052103  DH101:  .ASCIZ  ?EXPECTD (MMR0)  TESTNO  ERRORPC?
 2727  016326  020104  046450  051115
 2728  016334  024460  020040  042524
 2729  016342  052123  047516  020040
 2730  016350  051105  047522  050122
 2731  016356  000103
```

```
 2732  016360  044513  042120  032122  DH102:  .ASCIZ  ?KIPDR4  TESTNO  ERRORPC?
 2733  016366  020040  042524  052123
 2734  016374  047516  020040  051105
 2735  016402  047522  050122  000103
 2736  016410  044513  042120  033522  DH110:  .ASCIZ  ?KIPDR7  TESTNO  ERRORPC?
 2737  016416  020040  042524  052123
 2738  016424  047516  020040  051105
 2739  016432  047522  050122  000103
 2740  016440  042107  040520  042507  DH112:  .ASCIZ  ?GDPAGE  BDPAGE  TESTNO  ERRORPC?
 2741  016446  020040  042102  040520
 2742  016454  042507  020040  042524
 2743  016462  052123  047516  020040
 2744  016470  051105  047522  050122
 2745  016476  000103
 2746  016500  051524  050124  043501  DH113:  .ASCIZ  ?TSTPAGE CONTENT TESTNO  ERRORPC?
 2747  016506  020105  047503  052116
 2748  016514  047105  020124  042524
 2749  016522  052123  047516  020040
 2750  016530  051105  047522  050122
 2751  016536  000103
 2752  016540  054105  042520  052103  DH114:  .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
 2753  016546  020104  042522  042503
 2754  016554  053111  020104  042524
 2755  016562  052123  047516  020040
 2756  016570  051105  047522  050122
 2757  016576  000103
 2758  016600  054105  042520  052103  DH116:  .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERRORPC?
 2759  016606  020104  042522  042503
 2760  016614  053111  020104  042524
 2761  016622  052123  047516  020040
 2762  016630  051105  047522  050122
 2763  016636  000103
 2764  016640  046450  051115  024460  DH117:  .ASCIZ  ?(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC?
 2765  016646  020040  046450  051115
 2766  016654  024461  020040  046450
 2767  016662  051115  024462  020040
 2768  016670  042524  052123  047516
 2769  016676  020040  051105  047522
 2770  016704  050122  000103
 2771  016710  052123  050113  051124  DH120:  .ASCIZ  ?STKPTR  TESTNO  ERRORPC?
 2772  016716  020040  042524  052123
 2773  016724  047516  020040  051105
 2774  016732  047522  050122  000103
 2775  016740  042107  040504  040524  DH121:  .ASCIZ  ?GDDATA  STORED  TESTNO  ERRORPC?
 2776  016746  020040  052123  051117
 2777  016754  042105  020040  042524
 2778  016762  052123  047516  020040
 2779  016770  051105  047522  050122
 2780  016776  000103
 2781  017000  050050  053523  020051  DH122:  .ASCIZ  ?(PSW)   TESTNO  ERRORPC EXPECTING XXX340?
 2782  017006  020040  042524  052123
 2783  017014  047516  020040  051105
 2784  017022  047522  050122  020103
 2785  017030  054105  042520  052103
 2786  017036  047111  020107  054130
 2787  017044  031530  030064     000
```

```
 2788  017051     050  051520  024527  DH123:  .ASCIZ  ?(PSW)   TESTNO  ERRORPC?
 2789  017056  020040  052040  051505
 2790  017064  047124  020117  042440
 2791  017072  051122  051117  041520
 2792  017100     000
 2793  017101     124  051505  047124  DH127:  .ASCIZ  ?TESTNO ERRORPC?
 2794  017106  020117  051105  047522
 2795  017114  050122  000103
 2796  017120  046450  051115  024460  DH131:  .ASCIZ  ?(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC EXPECTING 020031?
 2797  017126  020040  046450  051115
 2798  017134  024461  020040  046450
 2799  017142  051115  024462  020040
 2800  017150  042524  052123  047516
 2801  017156  020040  051105  047522
 2802  017164  050122  020103  054105
 2803  017172  042520  052103  047111
 2804  017200  020107  031060  030060
 2805  017206  030463     000
 2806  017211     101  042104  042522  DH133:  .ASCIZ  ?ADDRESS EXPECT  RECEIVE  TESTNO  ERRPC?
 2807  017216  051523  042440  050130
 2808  017224  041505  020124  051040
 2809  017232  041505  044505  042526
 2810  017240  020040  042524  052123
 2811  017246  047516  020040  051105
 2812  017254  050122  000103
 2813
 2814
 2815  017260  042101  051104  051505  DH201:  .ASCIZ  ?ADDRESS TESTNO?
 2816  017266  020123  042524  052123
 2817  017274  047516     000
 2818  017277     101  042104  042522  DH202:  .ASCII  ?ADDRESS ADDRESS?<CRLF>
 2819  017304  051523  040440  042104
 2820  017312  042522  051523     200
 2821  017317     114  040517  042504          .ASCIZ  ?LOADED  JUSTREAD TESTNO?
 2822  017324  020104  045040  051525
 2823  017332  051124  040505  020104
 2824  017340  042524  052123  047516
 2825  017346     000
 2826  017347     101  042104  042522  DH203:  .ASCIZ  ?ADDRESS DATARED PATTERN COUNT   TESTNO?
 2827  017354  051523  042040  052101
 2828  017362  051101  042105  050040
 2829  017370  052101  042524  047122
 2830  017376  041440  052517  052116
 2831  017404  020040  052040  051505
 2832  017412  047124  000117
 2833                                  .EVEN
 2834
 2835
 2836  017416  001222                  DT1:    .WORD   CPUEXP
 2837  017420  001256  001262  001260  DT2:    .WORD   PCPUER,TESTNO,BADPC,0
 2838  017426  000000
 2839  017430  001236  001232  001240  DT3:    .WORD   PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0
 2840  017436  001242  001262  001260
 2841  017444  000000
 2842  017446  001246  001250  001252  DT5:    .WORD   PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
 2843  017454  001262  001260  000000
```

```
2844  017462  001224  001246  001250  DT6:    .WORD   MMEXP,PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
2845  017470  001252  001262  001260
2846  017476  000000
2847  017500  001156  001262  001116  DT7:    .WORD   $REG1,TESTNO,$ERRPC,0
2848  017506  000000
2849  017510  001160  001156  001262  DT11:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
2850  017516  001116  000000
2851  017522  001160  001262  001116  DT12:   .WORD   $REG2,TESTNO,$ERRPC,0
2852  017530  000000
2853  017532  001162  001160  001262  DT13:   .WORD   $REG3,$REG2,TESTNO,$ERRPC,0
2854  017540  001116  000000
2855  017544  001156                  DT16:   .WORD   $REG1
2856  017546  001160  001262  001116  DT15:   .WORD   $REG2,TESTNO,$ERRPC,0
2857  017554  000000
2858  017556  001274  001276  001262  DT17:   .WORD   ADDROR,ADRAND,TESTNO,ERRCNT,0
2859  017564  001300  000000
2860  017570  001154  001160  001262  DT20:   .WORD   $REG0,$REG2,TESTNO,$ERRPC,0
2861  017576  001116  000000
2862  017602  001274  001276  001264  DT21:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0
2863  017610  001266  001262  001300
2864  017616  000000
2865  017620  001274  001276  001270  DT22:   .WORD   ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,ERRCNT,0
2866  017626  001272  001264  001266
2867  017634  001262  001300  000000
2868  017642  001154  001160  001156  DT23:   .WORD   $REG0,$REG2,$REG1,TESTNO,$ERRPC,0
2869  017650  001262  001116  000000
2870  017656  001154  001262  001116  DT24:   .WORD   $REG0,TESTNO,$ERRPC,0
2871  017664  000000
2872  017666  001154  001156  001262  DT25:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2873  017674  001116  000000
2874  017700  001154  001156  001160  DT31:   .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2875  017706  001262  001116  000000
2876  017714  001154  001262  001116  DT35:   .WORD   $REG0,TESTNO,$ERRPC,0
2877  017722  000000
2878  017724  001154  001156  001160  DT36:   .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2879  017732  001262  001116  000000
2880  017740  001154  001156  001160  DT37:   .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2881  017746  001262  001116  000000
2882  017754  001172  001174  001262  DT40:   .WORD   $TMP1,$TMP2,TESTNO,$ERRPC,0
2883  017762  001116  000000
2884  017766  001172  001262  001116  DT41:   .WORD   $TMP1,TESTNO,$ERRPC,0
2885  017774  000000
2886  017776  001160  001162  001154  DT42:   .WORD   $REG2,$REG3,$REG0,TESTNO,$ERRPC,0
2887  020004  001262  001116  000000
2888  020012  001262  001116  000000  DT43:   .WORD   TESTNO,$ERRPC,0
2889  020020  001156  001262  001116  DT44:   .WORD   $REG1,TESTNO,$ERRPC,0
2890  020026  000000
2891  020030  001154  001262  001116  DT45:   .WORD   $REG0,TESTNO,$ERRPC,0
2892  020036  000000
2893  020040  172350  177760  001262  DT50:   .WORD   KIPAR4,SIZELO,TESTNO,$ERRPC,0
2894  020046  001116  000000
2895  020052  001156  001162  001262  DT51:   .WORD   $REG1,$REG3,TESTNO,$ERRPC,0
2896  020060  001116  000000
2897  020064  001262  001116  000000  DT53:   .WORD   TESTNO,$ERRPC,0
2898  020072  001154  001156  001262  DT55:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2899  020100  001116  000000
```

```
 2900  020104  177572  172310  001262  DT56:   .WORD   MMR0,KIPDR4,TESTNO,$ERRPC,0
 2901  020112  001116  000000
 2902  020116  001170  001262  001116  DT64:   .WORD   $TMP0,TESTNO,$ERRPC,0
 2903  020124  000000
 2904  020126  001224  001246  001262  DT65:   .WORD   MMEXP,PMMR0,TESTNO,$ERRPC,0
 2905  020134  001116  000000
 2906  020140  001246  001262  001116  DT66:   .WORD   PMMR0,TESTNO,$ERRPC,0
 2907  020146  000000
 2908  020150  001156  001246  001250  DT70:   .WORD   $REG1,PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
 2909  020156  001252  001262  001116
 2910  020164  000000
 2911  020166  001156  001246  001262  DT72:   .WORD   $REG1,PMMR0,TESTNO,$ERRPC,0
 2912  020174  001116  000000
 2913  020200  001246  001250  001252  DT73:   .WORD   PMMR0,PMMR1,PMMR2,$TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
 2914  020206  001170  001172  001174
 2915  020214  001262  001116  000000
 2916  020222  001246  001250  001252  DT75:   .WORD   PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
 2917  020230  001262  001116  000000
 2918  020236  001156  001252  001262  DT100:  .WORD   $REG1,PMMR2,TESTNO,$ERRPC,0
 2919  020244  001116  000000
 2920  020250  001160  001246  001262  DT101:  .WORD   $REG2,PMMR0,TESTNO,$ERRPC,0
 2921  020256  001116  000000
 2922  020262  001170  001262  001116  DT102:  .WORD   $TMP0,TESTNO,$ERRPC,0
 2923  020270  000000
 2924  020272  001162  001156  001262  DT112:  .WORD   $REG3,$REG1,TESTNO,$ERRPC,0
 2925  020300  001116  000000
 2926  020304  001162  001154  001262  DT113:  .WORD   $REG3,$REG0,TESTNO,$ERRPC,0
 2927  020312  001116  000000
 2928  020316  001160  001156  001262  DT114:  .WORD   $REG2,$REG1,TESTNO,$ERRORPC,0
 2929  020324  021576  000000
 2930  020330  001154  001156  001262  DT116:  .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
 2931  020336  001116  000000
 2932  020342  001246  001250  001252  DT117:  .WORD   PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
 2933  020350  001262  001116  000000
 2934  020356  001156  001262  001116  DT120:  .WORD   $REG1,TESTNO,$ERRPC,0
 2935  020364  000000
 2936  020366  001154  001262  001116  DT122:  .WORD   $REG0,TESTNO,$ERRPC,0
 2937  020374  000000
 2938  020376  001262  001116  000000  DT127:  .WORD   TESTNO,$ERRPC,0
 2939  020404  001246  001250  001252  DT131:  .WORD   PMMR0,PMMR1,PMMR2,TESTNO,$ERRPC,0
 2940  020412  001262  001116  000000
 2941  020420  001170  001172  001174  DT133:  .WORD   $TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
 2942  020426  001262  001116  000000
 2943
 2944  020434  001154  001262  000000  DT201:  .WORD   $REG0,TESTNO,0
 2945  020442  001154  001156  001262  DT202:  .WORD   $REG0,$REG1,TESTNO,0
 2946  020450  000000
 2947  020452  001154  001160  001164  DT203:  .WORD   $REG0,$REG2,$REG4,$REG1,TESTNO,0
 2948  020460  001156  001262  000000
 2949
 2950
 2951  020466      000              DF1:    .BYTE   0
 2952  020467      000      000     DF2:    .BYTE   0,0,0
 2953  020472      000      002     DF3:    .BYTE   0,2,0,0,0,0
 2954  020475      000      000
 2955  020500      000      000     DF5:    .BYTE   0,0,0,0,0
```

```
 2956  020503  000  000
 2957  020505  000  000  000  DF6:    .BYTE   0,0,0,0,0,0
 2958  020510  000  000  000
 2959  020513  000  000  000  DF7:    .BYTE   0,0,0
 2960  020516  000  000  000  DF11:   .BYTE   0,0,0,0
 2961  020521  000
 2962  020522  000  000  000  DF12:   .BYTE   0,0,0
 2963  020525  000  000  000  DF13:   .BYTE   0,0,0,0
 2964  020530  000
 2965  020531  000  000  000  DF16:   .BYTE   0,0,0
 2966  020534  000  000  000  DF15:   .BYTE   0,0,0,0
 2967  020537  000
 2968  020540  000  000  000  DF17:   .BYTE   0,0,0,1
 2969  020543  001
 2970  020544  000  000  000  DF20:   .BYTE   0,0,0,0
 2971  020547  000
 2972  020550  000  000  000  DF21:   .BYTE   0,0,0,0,0,1
 2973  020553  000  000  001
 2974  020556  000  000  000  DF22:   .BYTE   0,0,0,0,0,0,0,1
 2975  020561  000  000  000
 2976  020564  000  001
 2977  020566  000  000  000  DF23:   .BYTE   0,0,0,0,0
 2978  020571  000  000
 2979  020573  000  000  000  DF24:   .BYTE   0,0,0
 2980  020576  000  000  000  DF25:   .BYTE   0,0,0,0
 2981  020601  000
 2982  020602  000  000  000  DF31:   .BYTE   0,0,0,0,0
 2983  020605  000  000
 2984  020607  000  000  000  DF35:   .BYTE   0,0,0
 2985  020612  000  000  000  DF36:   .BYTE   0,0,0,0,0
 2986  020615  000  000
 2987  020617  003  000  000  DF37:   .BYTE   3,0,0,0,0
 2988  020622  000  000
 2989  020624  004  004  000  DF40:   .BYTE   4,4,0,0
 2990  020627  000
 2991  020630  004  000  000  DF41:   .BYTE   4,0,0
 2992  020633  000  000  003  DF42:   .BYTE   0,0,3,0,0
 2993  020636  000  000
 2994  020640  000  000       DF43:   .BYTE   0,0
 2995  020642  000  000  000  DF44:   .BYTE   0,0,0
 2996  020645  003  000  000  DF45:   .BYTE   3,0,0
 2997  020650  004  004  000  DF50:   .BYTE   4,4,0,0
 2998  020653  000
 2999  020654  000  000  000  DF51:   .BYTE   0,0,0,0
 3000  020657  000
 3001  020660  000  000       DF53:   .BYTE   0,0
 3002  020662  000  000  000  DF55:   .BYTE   0,0,0,0
 3003  020665  000
 3004  020666  000  000  000  DF56:   .BYTE   0,0,0,0
 3005  020671  000
 3006  020672  000  000  000  DF64:   .BYTE   0,0,0
 3007  020675  000  000  000  DF65:   .BYTE   0,0,0,0
 3008  020700  000
 3009  020701  000  000  000  DF66:   .BYTE   0,0,0
 3010  020704  000  000  000  DF70:   .BYTE   0,0,0,0,0,0
 3011  020707  000  000  000
```

```
3012  020712  000     000     000  DF72:   .BYTE   0,0,0,0
3013  020715  000
3014  020716  000     000     000  DF73:   .BYTE   0,0,0,0,0,0,0,0
3015  020721  000     000     000
3016  020724  000     000
3017  020726  000     000     000  DF75:   .BYTE   0,0,0,0,0
3018  020731  000     000
3019  020733  000     000     000  DF100:  .BYTE   0,0,0,0
3020  020736  000
3021  020737  000     000     000  DF101:  .BYTE   0,0,0,0
3022  020742  000
3023  020743  000     000     000  DF102:  .BYTE   0,0,0
3024  020746  000     000     000  DF112:  .BYTE   0,0,0,0
3025  020751  000
3026  020752  000     000     000  DF113:  .BYTE   0,0,0,0
3027  020755  000
3028  020756  000     000     000  DF114:  .BYTE   0,0,0,0
3029  020761  000
3030  020762  000     000     000  DF116:  .BYTE   0,0,0,0
3031  020765  000
3032  020766  000     000     000  DF117:  .BYTE   0,0,0,0,0
3033  020771  000     000
3034  020773  000     000     000  DF120:  .BYTE   0,0,0
3035  020776  000     000     000  DF122:  .BYTE   0,0,0
3036  021001  000     000          DF127:  .BYTE   0,0
3037  021003  000     000     000  DF131:  .BYTE   0,0,0,0,0
3038  021006  000     000
3039
3040
3041
3042  021010  000     000          DF201:  .BYTE   0,0
3043  021012  000     000     000  DF202:  .BYTE   0,0,0
3044  021015  000     000     000  DF203:  .BYTE   0,0,0,0,0
3045  021020  000     000
3046                                .EVEN
3047
3048
3049                        ;;**********************************************************************
3050
3051                        .SBTTL  END OF PASS ROUTINE
3052
3053                        ;*INCREMENT THE PASS NUMBER ($PASS)
3054                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
3055                        ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
3056                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
3057                        ;*IF SW12=1 INHIBIT TRACE TRAP
3058                        ;*IF THERES A MONITOR GO TO IT
3059                        ;*IF THERE ISN'T JUMP TO LOOP
3060
3061  021022                $EOP:
3062  021022  000240               NOP
3063  021024  005037  001102       CLR     $TSTNM          ;;ZERO THE TEST NUMBER
3064  021030  005037  001204       CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
3065  021034  005237  001100       INC     $PASS           ;;INCREMENT THE PASS NUMBER
3066  021040  042737  100000  001100  BIC  #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
3067  021046  005327               DEC     (PC)+           ;;LOOP?
```

```
 3068  021050  000001
 3069  021052  003072
 3070  021054  012737
 3071  021056  000001
 3072  021060  021050
 3073  021062  104400  021070
 3074  021066  000407
 3075
 3076  021106
 3077  021106  013746  001100
 3078
 3079  021112  104410
 3080  021114  104400  021122
 3081  021120  000421
 3082
 3083  021164
 3084  021164  013746  001112
 3085
 3086  021170  104410
 3087  021172  104400  001215
 3088  021176  005037  001112
 3089  021202  013700  000042
 3090  021206  001414
 3091  021210  005046
 3092  021212  012746  021220
 3093  021216  000427
 3094
 3095  021220
 3096  021220  013700  000042
 3097  021224  001405
 3098  021226  000005
 3099  021230  004710
 3100  021232  000240
 3101  021234  000240
 3102  021236  000240
 3103  021240
 3104  021240  013746  177776
 3105  021244  042716  000020
 3106  021250  032737  010000  177570
 3107  021256  001005
 3108  021260  005137  021304
 3109  021264  100402
 3110  021266  052716  000020
 3111  021272  012746  021300
 3112  021276  000002
 3113
 3114
 3115  021300
 3116  021300  000137  040264
 3117  021304  000000
 3118  021306     377     377     000
 3119        021312
 3120
 3121
 3122
 3123

$EOPCT: .WORD   1
        BGT     $DOAGN          ;;YES
        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
$ENDCT: .WORD   1
        $EOPCT
        TYPE    ,65$            ;;TYPE ASCIZ STRING
        BR      64$             ;;GET OVER THE ASCIZ
;;65$:  .ASCIZ  <12><15>/END PASS #/
64$:
        MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                ;;TYPE PASS NUMBER
        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    ,67$            ;;TYPE ASCIZ STRING
        BR      66$             ;;GET OVER THE ASCIZ
;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
66$:
        MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                ;;TOTAL NUMBER OF ERRORS
        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
        CLR     $ERTTL          ;;CLEAR ERROR TOTAL
$GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
        CLR     -(SP)           ;;INSURE THE 'T' BIT IS CLEAR
        MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
        BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                ;;WITH A CLEARED 'T' BIT
$CLR.T:
        MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
        BEQ     $DOAGN          ;;RETURN ADDRESS
        RESET                   ;;CLEAR THE WORLD
$ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
        NOP                     ;;SAVE ROOM
        NOP                     ;;FOR
        NOP                     ;;ACT11
$DOAGN:
        MOV     @#PS,-(SP)      ;;PUT THE PS ON THE STACK AND
        BIC     #20,(SP)        ;;CLEAR THE 'T' BIT
        BIT     #BIT12,@#SWR    ;;RUN WITH TRACE TRAP?
        BNE     1$              ;;BR IF NO
        COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
        BMI     1$              ;;BR IF NO
        BIS     #20,(SP)        ;;SET TRACE TRAP
1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
$RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                ;;AN 'RTT' IF 'RTT' IS A LEGAL
                                ;;INSTRUCTION
$LOOP:
        JMP     @#LOOP          ;;RETURN
$TBIT:  .WORD   0               ;;'T' BIT STATE INDICATOR
$ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
        .EVEN
;;**************************************************************

.SBTTL  SCOPE HANDLER ROUTINE
```

```
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<6:0>
                                        ;*CALL
                                        ;*      SCOPE            ;;SCOPE=IOT

021312                                  $SCOPE:
021312  005037  001312                          CLR     RETRY           ;CLEAR THE RETRY FLAG BEFORE THIS TEST
021316  005037  001300                          CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
021322  006137  177570                          ROL     @#SWR           ;;LOOP ON PRESENT TEST?
021326  100514                                  BMI     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
021330  000416                          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
021332  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
021336  012737  021356  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
021344  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
021350  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
021354  000466                                  BR      $SVLAD          ;;GO TO THE NEXT TEST
021356  022626                          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
021360  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
021364  000426                                  BR      7$              ;;LOOP ON THE PRESENT TEST
021366                                  6$:;#####END OF CODE FOR THE XOR TESTER#####
021366  032737  000400  177570                  BIT     #BIT08,@#SWR    ;;LOOP ON SPEC. TEST?
021374  001407                                  BEQ     2$              ;;BR IF NO
021376  013746  177570                          MOV     @#SWR,-(SP)     ;;SET DESIRED TEST NUM. FROM SWR
021402  042716  000200                          BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
021406  122637  001102                          CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
021412  001462                                  BEQ     $OVER           ;;BR IF YES
021414  105737  001103                  2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
021420  001421                                  BEQ     3$              ;;BR IF NO
021422  123737  001115  001103                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
021430  101015                                  BHI     3$              ;;BR IF NO
021432  032737  001000  177570                  BIT     #BIT09,@#SWR    ;;LOOP ON ERROR?
021440  001404                                  BEQ     4$              ;;BR IF NO
021442  013737  001110  001106          7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
021450  000443                                  BR      $OVER
021452  105037  001103                  4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
021456  005037  001204                          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
021462  000415                                  BR      1$              ;;ESCAPE TO THE NEXT TEST
021464  032737  004000  177570          3$:     BIT     #BIT11,@#SWR    ;;INHIBIT ITERATIONS?
021472  001011                                  BNE     1$              ;;BR IF YES
021474  005737  001100                          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
021500  001406                                  BEQ     1$              ;;      INHIBIT ITERATIONS
021502  005237  001104                          INC     $ICNT           ;;INCREMENT ITERATION COUNT
021506  023737  001204  001104                  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
021514  002021                                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
021516  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
021524  013737  021574  001204                  MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
021532  105237  001102                  $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
021536  011637  001106                          MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
```

```
3180  021542  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
3181  021546  005037  001206                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3182  021552  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3183  021560  013737  001102  177570  $OVER:  MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER
3184  021566  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
3185  021572  000002                          RTI                     ;;FIXES PS
3186  021574  000310                  $MXCNT: 200.                    ;;MAX. NUMBER OF ITERATIONS
3187                                  ;;**************************************************************************
3188
3189                                  .SBTTL  ERROR HANDLER ROUTINE
3190
3191                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3192                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3193                                  ;*AND GO TO ERTYPE ON ERROR
3194                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3195                                  ;*SW15=1        HALT ON ERROR
3196                                  ;*              HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
3197                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
3198                                  ;*SW10=1        BELL ON ERROR
3199                                  ;*SW09=1        LOOP ON ERROR
3200                                  ;*CALL
3201                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3202
3203  021576                          $ERROR:
3204  021576  113737  001102  001262          MOVB    $TSTNM,TESTNO   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
3205  021604  005237  001300                  INC     ERRCNT          ;KEEP COUNT OF MULTIPLE ERRORS
3206  021610  010037  001154                  MOV     R0,$REG0        ;SAVE R0
3207  021614  010137  001156                  MOV     R1,$REG1        ;SAVE R1
3208  021620  010237  001160                  MOV     R2,$REG2        ;SAVE R2
3209  021624  010337  001162                  MOV     R3,$REG3        ;SAVE R3
3210  021630  010437  001164                  MOV     R4,$REG4        ;SAVE R4
3211  021634  010537  001166                  MOV     R5,$REG5        ;SAVE R5
3212  021640  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
3213  021644  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
3214  021646  013737  001102  177570          MOV     $TSTNM,@#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
3215  021654  005737  177570                  TST     @#SWR           ;;HALT ON ERROR = 1?
3216  021660  100001                          BPL     8$              ;;BRANCH IF NO
3217  021662  000000                          HALT                    ;;YES--HALT
3218  021664  032737  002000  177570  8$:     BIT     #BIT10,@#SWR    ;;BELL ON ERROR?
3219  021672  001402                          BEQ     1$              ;;NO - SKIP
3220  021674  104400  001210                  TYPE    ,$BELL          ;;RING BELL
3221  021700  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
3222  021704  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
3223  021710  162737  000002  001116          SUB     #2,$ERRPC
3224  021716  117737  157174  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
3225  021724  032737  020000  177570          BIT     #BIT13,@#SWR    ;;SKIP TYPEOUT IF SET
3226  021732  001004                          BNE     2$              ;;SKIP TYPEOUTS
3227  021734  004737  022070                  JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
3228  021740  104400  001215                  TYPE    ,$CRLF
3229  021744  005737  177570          2$:     TST     @#SWR           ;;HALT ON ERROR
3230  021750  100001                          BPL     9$              ;;SKIP IF CONTINUE
3231  021752  000000                          HALT                    ;;HALT ON ERROR!
3232  021754  022737  021230  000042  9$:     CMP     #$ENDAD,42      ;;ACT-11?
3233  021762  001001                          BNE     3$              ;;BRANCH IF NO
3234  021764  000000                          HALT                    ;;YES
3235  021766  032737  001000  177570  3$:     BIT     #BIT09,@#SWR    ;;LOOP ON ERROR SWITCH SET?
```

```
3236  021774  001402                    BEQ     4$              ;;BR IF NO
3237  021776  013716  001110            MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
3238  022002  005737  001206    4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
3239  022006  001402                    BEQ     5$              ;;BR IF NONE
3240  022010  013716  001206            MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
3241  022014                    5$:
3242  022014  032737  001000  177570    BIT     #SW9,SWR        ;IS THE LOOP ON ERROR SWITCH UP?
3243  022022  001421                    BEQ     EREXIT          ;BRANCH IF NOT LOOPING ON ERROR
3244  022024  005037  177766            CLR     CPUERR          ;CLEAR CPU ERROR REGISTER
3245  022030  012737  177777  177744    MOV     #-1,MEMERR      ;CLEAR MEMORY ERROR REGISTER
3246  022036  042737  176776  177572    BIC     #176776,MMR0    ;CLEAR MEMORY MANAGEMENT REG 0
3247                                                            ;LEAVE BIT0 AND BIT9 UNCHANGED
3248  022044  012737  177777  024520    MOV     #-1,CPFLAG      ;RE-INITIALIZE CP TRAP FLAG
3249  022052  012737  177777  024624    MOV     #-1,PAFLAG      ;RE-INITIALIZE PARITY TRAP FLAG
3250  022060  012737  177777  025056    MOV     #-1,MMFLAG      ;RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
3251  022066  000002            EREXIT: RTI                     ;RETURN TO TEST AFTER ERROR
3252
3253
3254                              .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
```

```
3255
3256
3257                    ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
3258                    ;*      THE ERROR MESSAGES.  IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
3259                    ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE.  THE ERROR
3260                    ;*      TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FCR EACH
3261                    ;*      ENTRY, 'EM', 'DH', 'DT', 'DF'.  THE 'EM' POINTS TO THE ERROR
3262                    ;*      MESSAGE WHICH IS AN ASCIZ STRING.  THE 'DH' POINTS TO THE DATA
3263                    ;*      HEADER WHICH IS ANOTHER ASCIZ STRING.  THE 'DT' POINTS TO THE
3264                    ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
```

```
                        ;*      OF THE DATA TO BY TYPED.  THE FORMAT OF THIS DATA IS
                        ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
                        ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
                        ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
                        ;*      0       -16 BIT OCTAL FORMAT
                        ;*      1       -DECIMAL FORMAT
                        ;*      2       -22 BIT OCTAL FORMAT.  DATA IS LOWER 16 BITS OF THE
                        ;*               PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
                        ;*      3       -22 BIT OCTAL FORMAT.  DATA IS THE 16 BIT VIRTUAL
                        ;*               ADDRESS IN KERNEL I-SPACE.
                        ;*      4       -22 BIT OCTAL FORMAT.  DATA IS A P.A.R. AND THE
                        ;*               OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
022070  010046          ERTYPE: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
022072  005000                  CLR     R0              ;CLEAR R0
022074  113700  001114          MOVB    @#$ITEMB,R0     ;PUT ITEM NUMBER IN R0
022100  001004                  BNE     1$              ;BRANCH IF IT IS NON-ZERO
022102  013746  001116          MOV     $ERRPC,-(KSP)   ;PUT ERROR PC ON STACK FOR TYPING
022106  104402                  TYPOC                   ;TYPE FAILING PC
022110  000551                  BR      13$             ;GO TO RETURN
022112  005300          1$:     DEC     R0              ;ADJUST ITEM NUMBER TO BE A POINTER
022114  072027  000003          ASH     #3,R0           ;LEFT SHIFT ITEM NO. 3 PLACES
022120  100024                  BPL     22$             ;BRANCH IF ITEM #LESS THAN 200
022122  032700  001000          BIT     #BIT9,R0        ;SEE IF ITEM # WAS 3XX
022126  001415                  BEQ     21$             ;BRANCH IF ITEM # WAS 2XX
                                                        ;AND TYPE ERROR MESSAGE
022130  032737  000200  177570  BIT     #SW7,@#SWR      ;SEE IF SWITCH 7 IS UP
022136  001404                  BEQ     20$             ;BRANCH IF SWITCH IS NOT UP
                                                        ;AND TYPE DATA, ON MULTIPLE ERRORS
022140  062766  000004  000002  ADD     #4,2(KSP)       ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
                                                        ;INHIBIT MULTIPLE ERROR TYPEOUTS
022146  000532                  BR      13$             ;BRANCH TO EXIT
022150  042700  177000  20$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
022154  062700  002712          ADD     #ER200+4,R0     ;POINT TO DATA TABLE ENTRY
022160  000424                  BR      5$              ;GO TYPE DATA TABLE
022162  042700  177000  21$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
022166  062700  001330          ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
                                                        ;ITEM 1 AND ITEM 201
                        ;;                      GET POINTER TO ERROR MESSAGE AND TYPE IT
                        ;;                      IF THE POINTER IS NOT ZERO
022172  062700  001356  22$:    ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
022176  012037  022206          MOV     (R0)+,2$        ;PUT MESSAGE POINTER IN TYPE STATEMENT
022202  001404                  BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
022204  104400                  TYPE                    ;TYPE ERROR MESSAGE
022206  000000          2$:     .WORD   0               ;POINTER TO ERROR MESSAGE
022210  104400  001215          TYPE    ,$CRLF          ;TYPE CRLF
                        ;;                      GET THE POINTER TO THE DATA HEADER AND
                        ;;                      TYPE IT IF THE POINTER IS NOT ZERO
022214  012037  022224  3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
022220  001404                  BEQ     5$              ;BRANCH IF NO DATA HEADER
022222  104400                  TYPE                    ;TYPE THE DATA HEADER
022224  000000          4$:     .WORD   0               ;POINTER TO DATA HEADER
022226  104400  001215          TYPE    ,$CRLF          ;TYPE CRLF
                        ;;                      THIS IS THE START OF THE DATA OUTPUT IF THE
                        ;;                      DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
                        ;;                      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
                        ;;                      THE DATA WORDS.
```

```
3321  022232  010146              5$:   MOV   R1,-(KSP)        ;SAVE R1 ON THE STACK
3322  022234  012001                    MOV   (R0)+,R1         ;PUT DATA TABLE POINTER IN R1
3323  022236  001475                    BEQ   12$              ;BRANCH IF NO DATA TABLE
3324  022240  012000                    MOV   (R0)+,R0         ;PICK UP DATA FORMAT POINTER
3325  022242  105710              6$:   TSTB  (R0)             ;IS THIS WORD OCTAL
3326  022244  001003                    BNE   7$               ;BRANCH IF NOT 16-BIT OCTAL
3327                                ;:                  THIS IS 16 BIT OCTAL FORMAT (DF = 0)
3328  022246  013146                    MOV   @(R1)+,-(KSP)    ;PUT WORD ON STACK FOR TYPING
3329  022250  104402                    TYPOC                  ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
3330  022252  000461                    BR    11$              ;GET READY FOR NEXT WORD
3331  022254  122710  000001    7$:    CMPB  #1,(R0)          ;IS THE WORD DECIMAL
3332  022260  001003                    BNE   8$               ;BRANCH IF NOT DECIMAL
3333                                ;:                  THIS IS DECIMAL FORMAT (DF = 1)
3334  022262  013146                    MOV   @(R1)+,-(KSP)    ;PUT WORD ON STACK FOR TYPING
3335  022264  104410                    TYPDS                  ;TYPE THE WORD ON STACK AS DECIMAL
3336  022266  000453                    BR    11$              ;GET READY FOR NEXT WORD
3337  022270  122710  000002    8$:    CMPB  #2,(R0)          ;IS WORD 22-BIT PHYSICAL ADDRESS
3338  022274  001012                    BNE   9$               ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
3339                                ;:                  THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
3340  022276  012146                    MOV   (R1)+,-(KSP)     ;PUT ADDR OF LOW WORD ON STACK
3341  022300  004737  023772            JSR   PC,$DB2O         ;CONVERT NUMBER TO OCTAL ASCIZ
3342  022304  062716  000003            ADD   #3,(KSP)                 ;ONLY WANT 8 DIGITS
3343  022310  012637  022316            MOV   (KSP)+,30$       ;PUT POINTER AFTER 'TYPE' CALL
3344  022314  104400                    TYPE                   ;TYPE ASCIZ STRING
3345  022316  000000            30$:    .WORD 0                ;WORD HOLDS POINTER TO ASCIZ STRING
3346  022320  000436                    BR    11$              ;GET READY FOR NEXT WORD
3347  022322  122710  000003    9$:    CMPB  #3,(R0)          ;IS THIS A 16-BIT VIRTUAL ADDRESS
3348  022326  001004                    BNE   10$              ;BRANCH IF NOT 16-BIT VIRT. ADDR.
3349                                ;:                  THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
3350                                ;:                  KERNEL I-SPACE ASSUMED.  (DF = 3)
3351  022330  013146                    MOV   @(R1)+,-(KSP)    ;PUT 16-BIT VIRTUAL ADDR ON STACK
3352  022332  004737  022444            JSR   PC,TYPVAD        ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
3353  022336  000427                    BR    11$              ;GET READY FOR NEXT WORD
3354                                ;:                  THIS IS FORMAT 4.  DATA WORD IS A P.A.R.
3355                                ;:                  OUTPUT WILL BE 22-BIT. PAR LEFT SHIFTED 6.
3356  022340  010246            10$:    MOV   R2,-(KSP)        ;SAVE R2 ON THE STACK
3357  022342  010346                    MOV   R3,-(KSP)        ;SAVE R3 ON THE STACK
3358  022344  013103                    MOV   @(R1)+,R3        ;LOAD DATA WORD INTO R3
3359  022346  005002                    CLR   R2               ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
3360  022350  073227  000006            ASHC  #6,R2            ;LEFT SHIFT <R2:R3> 6 PLACES
3361  022354  010237  001230            MOV   R2,PADRSH        ;STORE UPPER BITS OF ADDRESS
3362  022360  010337  001226            MOV   R3,PADRSL        ;STORE LOWER 16 BITS OF ADDRESS
3363  022364  012746  001226            MOV   #PADRSL,-(KSP)   ;PUT ADDRESS OF LOWER 16 BITS ON STACK
3364  022370  004737  023772            JSR   PC,$DB2O         ;CONVERT TWO WORDS TO ASCIZ
3365  022374  062716  000003            ADD   #3,(KSP)         ;I ONLY WANT 8 DIGITS
3366  022400  012637  022406            MOV   (KSP)+,31$       ;PUT POINTER AFTER TYPE CALL
3367  022404  104400                    TYPE                   ;POINTER TO ASCIZ STRING FOLLOWS
3368  022406  000000            31$:    .WORD 0                ;POINTER TO ASCIZ STRING
3369  022410  011603                    MOV   (KSP),+R3        ;RESTORE R3 FROM STACK
3370  022412  012602                    MOV   (KSP)+,R2        ;RESTORE R2 FROM STACK
3371  022414  000400                    BR    11$              ;GET READY FOR NEXT WORD
3372  022416  005200            11$:    INC   R0               ;POINT TO NEXT FORMAT BYTE
3373  022420  104400  022440            TYPE  ,32$             ;TYPE TWO SPACES
3374  022424  005711                    TST   (R1)             ;IS THERE ANOTHER WORD?
3375  022426  001401                    BEQ   12$              ;BRANCH IF ALL DONE
3376  022430  000704                    BR    6$               ;GO BACK FOR NEXT NUMBER
```

```
3377  022432  012601              12$:    MOV     (KSP)+,R1       ;RESTORE R1
3378  022434  012600              13$:    MOV     (KSP)+,R0       ;RESTORE R0
3379  022436  000207                      RTS     PC              ;RETURN TO ERROR ROUTINE
3380  022440  020040      000     32$:    .ASCIZ  ?  ?            ;TWO SPACES
3381          022444                      .EVEN
3382
3383
3384                              .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
3385                              ;*
3386                              ;*      THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
3387                              ;*      IS PUSHED ON THE KERNEL STACK.  THE V.A. IS THEN LOADED INTO
3388                              ;*      R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
3389                              ;*      CORRECT KERNEL I-SPACE PAR.  THE LOWER 12 BITS OF THE VIRTUAL
3390                              ;*      ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
3391                              ;*      AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
3392                              ;*      TO ASCIZ AND TYPED.
3393                              ;*
3394  022444  104412              TYPVAD: SAVREG                  ;SAVE ALL REGISTERS
3395  022446  016601  000002              MOV     2(KSP),R1       ;PUT VIRTUAL ADDR IN R1
3396  022452  005000                      CLR     R0              ;CLEAR R0 FOR CALCULATIONS
3397  022454  073027  000003              ASHC    #3,R0           ;LEFT SHIFT R0,R1 3 PLACES
3398  022460  006300                      ASL     R0              ;LEFT SHIFT R0 ONE MORE PLACE
3399  022462  006001                      ROR     R1              ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
3400  022464  006001                      ROR     R1              ;RIGHT SHIFT R1
3401  022466  006001                      ROR     R1              ;RIGHT SHIFT R1
3402  022470  062700  172340              ADD     #KIPAR0,R0      ;FORM DESIRED PAR ADDR IN R0
3403  022474  011003                      MOV     (R0),R3         ;PUT CONTENTS OF PAR IN R3
3404  022476  005002                      CLR     R2              ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
3405  022500  073227  000006              ASHC    #6,R2           ;LEFT SHIFT <R2,R3> 6 PLACES
3406  022504  060103                      ADD     R1,R3           ;ADD OFFSET IN R1 TO BASE IN R3
3407  022506  005502                      ADC     R2              ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
3408  022510  010237  001230              MOV     R2,PADRSH       ;PUT UPPER 6 BITS OF ADDR IN CORE
3409  022514  010337  001226              MOV     R3,PADRSL       ;PUT LOWER 16 BITS OF ADDR IN CORE
3410  022520  012746  001226              MOV     #PADRSL,-(KSP)  ;PUT POINTER TO LOWER 16 BITS ON STACK
3411  022524  004737  023772              JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
3412  022530  062716  000003              ADD     #3,(KSP)        ;ONLY TYPE 8 DIGITS
3413  022534  012637  022542              MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE INST
3414  022540  104400                      TYPE                    ;TYPE THE 22-BIT VIRTUAL ADDRESS
3415  022542  000000              3$:     .WORD   0               ;THIS WORD HOLDS THE POINTER TO
3416                                                              ;THE ASCIZ STRING
3417  022544  104414                      RESREG                  ;RESTORE ALL THE REGISTERS
3418  022546  012616                      MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDR ON STACK
3419  022550  000207                      RTS     PC              ;RETURN TO ERROR HANDLER
3420
3421
3422                              ;;*****************************************************************
3423
3424                              .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
3425
3426                              ;*SAVE R0-R5
3427                              ;*CALL:
3428                              ;*      SAVREG
3429                              ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3430                              ;*
3431                              ;*TOP---(+16)
3432                              ;*  +2---(+18)
```

```
                                ;* +4---R5
                                ;* +6---R4
                                ;* +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0
                                
022552                          $SAVREG:
022552  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022554  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022556  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022560  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022562  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
022564  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022566  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
022572  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
022576  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
022602  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
022606  000002                          RTI
                                
                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
022610                          $RESREG:
022610  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
022614  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
022620  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
022624  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
022630  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
022632  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
022634  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
022636  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
022640  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
022642  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022644  000002                          RTI
                                ;;*****************************************************************
                                
                                .SBTTL  TYPE ROUTINE
                                
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*
                                ;*2) USING A JSR INSTRUCTION
                                ;*      MOV     PS,-(SP)        ;;PUSH PROCESSOR STATUS WORD ON THE STACK
                                ;*      JSR     PC,$TYPE        ;;CALL TYPE ROUTINE
                                ;*      MESADDR                 ;;FIRST ADRESS OF MESSAGE
```

```
3489
3490  022646  105737  001151          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
3491  022652  100002                          BPL     1$              ;;BR IF YES
3492  022654  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
3493  022656  000407                          BR      3$              ;;LEAVE
3494  022660  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
3495  022662  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
3496  022666  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3497  022670  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
3498  022672  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
3499  022674  012600                          MOV     (SP)+,R0        ;;RESTORE R0
3500  022676  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
3501  022702  000002                          RTI                     ;;RETURN
3502  022704  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
3503  022710  001424                          BEQ     8$
3504  022712  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT
3505  022716  001004                          BNE     5$
3506  022720  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
3507  022722  104400  001215                  TYPE    ,$CRLF
3508  022726  000757                          BR      2$              ;;GET NEXT CHARACTER
3509  022730  004737  023006          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
3510  022734  123726  001150          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
3511  022740  001352                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
3512  022742  013746  001146                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
3513                                                                  ;;AND THE NULL CHAR.
3514  022746  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
3515  022752  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
3516  022754  004737  023006                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
3517  022760  000772                          BR      7$              ;;LOOP
3518
3519                                  ;;HORIZONTAL TAB PROCESSOR
3520
3521  022762  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
3522  022766  004737  023006          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
3523  022772  132737  000007  023052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
3524  023000  001372                          BNE     9$              ;;TAB STOP
3525  023002  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
3526  023004  000730                          BR      2$              ;;GET NEXT CHARACTER
3527  023006  105777  156130          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3528  023012  100375                          BPL     $TYPEC
3529  023014  116677  000002  156122          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3530  023022  122766  000015  000002          CMPB    #CR,2(SP)       ;;BRANCH IF
3531  023030  001003                          BNE     1$              ;;NOT <CR>
3532  023032  105037  023052                  CLRB    $CHARCNT        ;;
3533  023036  000406                          BR      $TYPEX          ;;EXIT
3534  023040  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;BRANCH IF
3535  023046  001402                          BEQ     $TYPEX          ;;<LF>
3536  023050  105227                          INCB    (PC)+           ;;INC SPACE
3537  023052  000000          $CHARCNT:.WORD  0                       ;;COUNT
3538  023054  000207          $TYPEX: RTS     PC
3539
3540                                  ;;******************************************************************
3541
3542                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3543
3544                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
```

```
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
023056  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
023062  116637  000001  023301          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
023070  112637  023303                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
023074  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
023100  000406                          BR      $TYPON
023102  112737  000001  023301  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
023110  112737  000006  023303          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
023116  112737  000005  023300  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
023124  010346                          MOV     R3,-(SP)        ;;SAVE R3
023126  010446                          MOV     R4,-(SP)        ;;SAVE R4
023130  010546                          MOV     R5,-(SP)        ;;SAVE R5
023132  113704  023303                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
023136  005404                          NEG     R4
023140  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
023144  110437  023302                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
023150  113704  023301                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
023154  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
023160  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
023162  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO ''C''
023164  000404                          BR      3$              ;;GO DO MSB
023166  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
023170  006105                          ROL     R5
023172  006105                          ROL     R5
023174  010503                          MOV     R5,R3
023176  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
023200  105337  023302                  DECB    $OMODE          ;;TYPE THIS DIGIT?
023204  100016                          BPL     7$              ;;BR IF NO
023206  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
023212  001002                          BNE     4$              ;;TEST FOR 0
023214  005704                          TST     R4              ;;SUPPRESS THIS 0?
023216  001403                          BEQ     5$              ;;BR IF YES
023220  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
023222  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
023226  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
023232  110337  023276                  MOVB    R3,8$           ;;SAVE FOR TYPING
```

```
3601  023236  104400  023276                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
3602  023242  105337  023300          7$:     DECB    $OCNT           ;;COUNT BY 1
3603  023246  003347                          BGT     2$              ;;BR IF MORE TO DO
3604  023250  002402                          BLT     6$              ;;BR IF DONE
3605  023252  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3606  023254  000744                          BR      2$              ;;GO DO THE LAST DIGIT
3607  023256  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
3608  023260  012604                          MOV     (SP)+,R4        ;;RESTORE R4
3609  023262  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3610  023264  016666  000002  000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3611  023272  012616                          MOV     (SP)+,(SP)
3612  023274  000002                          RTI                     ;;RETURN
3613  023276     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3614  023277     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3615  023300     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3616  023301     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3617  023302  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3618                          ;;**************************************************************
3619
3620                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3621
3622                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3623                          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3624                          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3625                          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3626                          ;*REPLACED WITH SPACES.
3627                          ;*CALL:
3628                          ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
3629                          ;*      TYPDS                   ;;GO TO THE ROUTINE
3630
3631  023304                  $TYPDS:
3632  023304  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3633  023306  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3634  023310  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3635  023312  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3636  023314  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3637  023316  012746  020200          MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
3638  023322  016605  000020          MOV     20(SP),R5       ;;GET THE INPUT NUMBER
3639  023326  100004                  BPL     1$              ;;BR IF INPUT IS POS.
3640  023330  005405                  NEG     R5              ;;MAKE THE BINARY NUMBER POS.
3641  023332  112766  000055  000001 MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
3642  023340  005000          1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
3643  023342  012703  023520          MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
3644  023346  112723  000040          MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
3645  023352  005002          2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
3646  023354  016001  023510          MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
3647  023360  160105          3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
3648  023362  002402                  BLT     4$              ;;BR IF DONE
3649  023364  005202                  INC     R2              ;;INCREASE THE BCD DIGIT BY 1
3650  023366  000774                  BR      3$
3651  023370  060105          4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
3652  023372  005702                  TST     R2              ;;CHECK IF BCD DIGIT=0
3653  023374  001002                  BNE     5$              ;;FALL THROUGH IF 0
3654  023376  105716                  TSTB    (SP)            ;;STILL DOING LEADING 0'S?
3655  023400  100407                  BMI     7$              ;;BR IF YES
3656  023402  106316          5$:     ASLB    (SP)            ;;MSD?
```

```
3657  023404  103003                        BCC     6$                ;;BR IF NO
3658  023406  116663  000001  177777        MOVB    1(SP),-1(R3)      ;;YES--SET THE SIGN
3659  023414  052702  000060          6$:   BIS     #'0,R2            ;;MAKE THE BCD DIGIT ASCII
3660  023420  052702  000040          7$:   BIS     #' ,R2            ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3661  023424  110223                        MOVB    R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3662  023426  005720                        TST     (R0)+             ;;JUST INCREMENTING
3663  023430  020027  000010                CMP     R0,#10            ;;CHECK THE TABLE INDEX
3664  023434  002746                        BLT     2$                ;;GO DO THE NEXT DIGIT
3665  023436  003002                        BGT     8$                ;;GO TO EXIT
3666  023440  010502                        MOV     R5,R2             ;;GET THE LSD
3667  023442  000764                        BR      6$                ;;GO CHANGE TO ASCII
3668  023444  105726                  8$:   TSTB    (SP)+             ;;WAS THE LSD THE FIRST NON-ZERO?
3669  023446  100003                        BPL     9$                ;;BR IF NO
3670  023450  116663  177777  177776        MOVB    -1(SP),-2(R3)     ;;YES--SET THE SIGN FOR TYPING
3671  023456  105013                  9$:   CLRB    (R3)              ;;SET THE TERMINATOR
3672  023460  012605                        MOV     (SP)+,R5          ;;POP STACK INTO R5
3673  023462  012603                        MOV     (SP)+,R3          ;;POP STACK INTO R3
3674  023464  012602                        MOV     (SP)+,R2          ;;POP STACK INTO R2
3675  023466  012601                        MOV     (SP)+,R1          ;;POP STACK INTO R1
3676  023470  012600                        MOV     (SP)+,R0          ;;POP STACK INTO R0
3677  023472  104400  023520                TYPE    ,$DBLK            ;;NOW TYPE THE NUMBER
3678  023476  016666  000002  000004        MOV     2(SP),4(SP)       ;;ADJUST THE STACK
3679  023504  012616                        MOV     (SP)+,(SP)
3680  023506  000002                        RTI                       ;;RETURN TO USER
3681  023510  023420                  $DTBL:  10000.
3682  023512  001750                         1000.
3683  023514  000144                         100.
3684  023516  000012                         10.
3685  023520  000004                  $DBLK:  .BLKW   4
3686                                  ;;*****************************************************************
3687
3688                                  .SBTTL  TRAP DECODER
3689
3690                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
3691                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3692                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3693                                  ;*GO TO THAT ROUTINE.
3694
3695  023530  010046                  $TRAP:  MOV     R0,-(SP)          ;;SAVE R0
3696  023532  016600  000002                 MOV     2(SP),R0          ;;GET TRAP ADDRESS
3697  023536  005740                         TST     -(R0)             ;;BACKUP BY 2
3698  023540  111000                         MOVB    (R0),R0           ;;GET RIGHT BYTE OF TRAP
3699  023542  016000  023550                 MOV     $TRPAD(R0),R0     ;;INDEX TO TABLE
3700  023546  000200                         RTS     R0                ;;GO TO ROUTINE
3701
3702
3703                                  .SBTTL  TRAP TABLE
3704
3705                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3706                                  ;*BY THE ''TRAP'' INSTRUCTION.
3707
3708                                  ;       ROUTINE
3709                                  ;       -------
3710  023550                          $TRPAD:
3711  023550  022646                         $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
3712  023552  023102                         $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
```

```
3713  023554  023056                  $TYPOS  ;;CALL=TYPOS    TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3714  023556  023116                  $TYPON  ;;CALL=TYPON    TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3715  023560  023304                  $TYPDS  ;;CALL=TYPDS    TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
3716  023562  022552                  $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
3717  023564  022610                  $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
3718  023566  024112                  TBITOFF ;;CALL=TBITO    TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
3719  023570  024140-                 TBITRESTORE     ;;CALL=TBITR    TRAP+20(104420) THIS WILL RETURN THE T BIT TO PR
3720
3721                                  ;;*****************************************************************
3722
3723                                  .SBTTL  POWER DOWN AND UP ROUTINES
3724
3725                                  ;POWER DOWN ROUTINE
3726  023572  012737  023720  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
3727  023600  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
3728  023606  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3729  023610  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3730  023612  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3731  023614  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3732  023616  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3733  023620  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3734  023622  010637  023724                  MOV     SP,$SAVR6       ;;SAVE SP
3735  023626  012737  023640  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
3736  023634  000000                          HALT
3737  023636  000776                          BR      .-2             ;;HANG UP
3738
3739                                  ;POWER UP ROUTINE
3740  023640  013706  023724  $PWRUP: MOV     $SAVR6,SP       ;;GET SP
3741  023644  005037  023724          CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
3742  023650  005237  023724  1$:     INC     $SAVR6          ;;WAIT FOR THE INC
3743  023654  001375                  BNE     1$              ;;OF  WORD
3744  023656  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
3745  023660  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
3746  023662  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3747  023664  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
3748  023666  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3749  023670  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
3750  023672  012737  023572  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3751  023700  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
3752  023706  104400                          TYPE                    ;;REPORT THE POWER FAILURE
3753  023710  023726  $PWRMG:  .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
3754  023712  012716          MOV     (PC)+,(SP)      ;;RESTART AT START
3755  023714  040076  $PWRAD:  .WORD   START           ;;RESTART ADDRESS
3756  023716  000002          RTI
3757  023720  000000  $ILLUP:  HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
3758  023722  000776          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
3759  023724  000000  $SAVR6:  0                       ;;PUT THE SP HERE
3760  023726  006412  047520  042527  PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
3761  023734  020122  040506  046111
3762  023742  051125  026105  051040
3763  023750  051505  040524  052122
3764  023756  047111  020107  051120
3765  023764  043517  040522  000115
3766                                          .EVEN
3767
3768                                  ;;*****************************************************************
```

```
                                                    H 6
 3769
 3770                                   .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
 3771
 3772                                   ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
 3773                                   ;*UNSIGNED OCTAL ASCIZ NUMBER.
 3774                                   ;*CALL
 3775                                   ;*       MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
 3776                                   ;*       JSR     PC,@#$DB20      ;;CALL THE ROUTINE
 3777                                   ;*       RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
 3778
 3779
 3780  023772  104412                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
 3781  023774  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
 3782  024000  012705  024111                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
 3783  024004  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
 3784  024010  012703  177770                  MOV     #^C7,R3         ;;MASK
 3785  024014  012100                          MOV     (R1)+,R0        ;;LOWER WORD
 3786  024016  012101                          MOV     (R1)+,R1        ;;HIGH WORD
 3787  024020  005002                          CLR     R2              ;;TERMINATOR
 3788  024022  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
 3789  024024  010002                          MOV     R0,R2           ;;GET THIS DIGIT
 3790  024026  005304                          DEC     R4              ;;COUNT THIS CHARACTER
 3791  024030  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
 3792  024032  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
 3793  024034  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
 3794  024036  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
 3795  024042  104414                          RESREG                  ;;RESTORE ALL REGISTERS
 3796  024044  000207                          RTS     PC              ;;RETURN TO USER
 3797  024046  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
 3798  024050  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
 3799  024052  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
 3800  024054  006001                          ROR     R1
 3801  024056  006000                          ROR     R0
 3802  024060  006001                          ROR     R1
 3803  024062  006000                          ROR     R0
 3804  024064  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
 3805  024066  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
 3806  024072  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
 3807  024074  000016                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
 3808
 3809
 3810                                   .SBTTL  ************** SUBROUTINES UNIQUE TO THIS PROGRAM ***************
 3811
 3812
 3813
 3814                                   .SBTTL  TURN OFF AND SAVE T-BIT
 3815                                   ;*
 3816                                   ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
 3817                                   ;*      USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
 3818                                   ;*      IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
 3819                                   ;*      PREVIOUS STATUS WHEN CONDITIONS WARRANT.
 3820                                   ;*
 3821                                           .EQUIV  BIT4,TBIT       ;T-BIT IS BIT04 IN PROC. STATUS
 3822  024112                          TBITOFF:
 3823  024112  032766  000020  000002          BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
 3824  024120  001406                          BEQ     1$              ;BRANCH TO EXIT IF IT IS NOT ON
```

```
3825  024122  016637  000002  001310          MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING T BIT
3826  024130  042766  000020  000002          BIC     #TBIT,2(KSP)    ;CLEAR T BIT IF IT IS ON
3827  024136  000006                  1$:     RTT                     ;RETURN TO PROGRAM
3828
3829
3830
3831                                  .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
3832                                  ;*
3833                                  ;*      THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
3834                                  ;*      USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
3835                                  ;*      BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR STATUS STORED
3836                                  ;*      IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
3837                                  ;*      AND DOES AN 'RTT'.
3838                                  ;*
3839  024140                          TBITRESTORE:
3840  024140  013766  001310  000002          MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
3841  024146  042737  000020  001310          BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN 'OLDPSW' SO THAT
3842                                                                  ;IT WON'T BE TURNED ON BY ACCIDENT
3843  024154  000006                          RTT                     ;RETURN TO PROGRAM AND INHIBIT
3844                                                                  ;T BIT TRAP AFTER THIS INSTRUCTION.
3845
3846
3847
3848                                  .SBTTL  CLEAR 16 PARS OR PDRS STARTING FROM ADDRESS IN R5
3849                                  ;;**************************************************************************
3850                                  ;*      SUBROUTINE CLRREG
3851                                  ;*
3852                                  ;*      THIS SUBROUTINE CLEARS 16 CONSECUTIVE MEMORY MANAGEMENT
3853                                  ;*      REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
3854                                  ;*      IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0,
3855                                  ;*      CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
3856                                  ;*      RETURNING.
3857                                  ;*      THE CALLING SEQUENCE IS:
3858                                  ;*      MOV     #KIPAR0,R5      ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
3859                                  ;*      JSR     PC,CLRREG       ;GO CLEAR ALL KERNEL PAR'S
3860                                  ;*
3861                                  ;;**************************************************************************
3862
3863  024156  010046                  CLRREG: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
3864  024160  012700  000020                  MOV     #20,R0          ;PUT COUNT IN R0
3865  024164  005025                  1$:     CLR     (R5)+           ;CLEAR PAR ORPDR POINTED TO BY R5
3866  024166  077002                          SOB     R0,1$           ;BRANCH BACK 15 DECIMAL TIMES.
3867  024170  012600                          MOV     (KSP)+,R0       ;RESTORE R0 FROM STACK
3868  024172  000207                          RTS     PC              ;RETURN TO TEST
3869
3870                                  .SBTTL  CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
3871                                  ;;**************************************************************************
3872                                  ;*      SUBROUTINE CLEANUP
3873                                  ;*
3874                                  ;*      THIS SUBROUTINE IS USED TO INITIALIZE ALL THE LOCATIONS THAT
3875                                  ;*      HOLD THE 'LOGICAL AND' AND 'LOGICAL OR' OF THE DATA AND ADDRESSES
3876                                  ;*      THAT FAILED DURING THE EXECUTION OF A TEST.
3877                                  ;*
3878                                  ;;**************************************************************************
3879
3880  024174                          CLEANUP:                        ;STARTING ADDRESS OF SUBROUTINE.
```

```
                                        CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
                                        CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
                                        CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
                                        MOV     #-1,R0          ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
                                        MOV     R0,DATAND       ;LOCATION FOR LOGICAL AND OF BAD DATA
                                        MOV     R0,ADRAND       ;LOCATION FOR LOGICAL AND OF ADDRESS
                                        MOV     R0,PATAND       ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
                                        RTS     PC              ;RETURN TO TEST

                                .SBTTL  P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
                                ;;*************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
                                ;*      REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS.  IT
                                ;*      KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
                                ;*      OUT.
                                ;;*************************************************************************
024232                          TIMEOUT:                        ;STARTING ADDRESS OF SUBROUTINE
024232  005227                          INC     (PC)+           ;INCREMENT ONE TIME GATE
024234  177777                  TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
024236  001401                          BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
024240  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE
                                                                ;STACK AND THE FIRST ERROR CONDITION
                                                                ;IS PROBABLY STILL LOCKED UP .
024242  012637  001304          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
024246  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
024252  013737  177766  001256          MOV     @#CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
024260  013737  001256  177766          MOV     PCPUER,@#CPUERR ;CLEAR CPU ERROR REGISTER
024266  023737  001256  001222          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTEED CONDITION CAME UP.
024274  001402                          BEQ     1$              ;BRANCH IF IT WAS A TIMEOUT
024276  104001                          ERROR   1               ;NOT RIGHT CONDITION
024300  000414                          BR      3$              ;BRANCH TO EXIT
024302  050037  001274          1$:     BIS     R0,ADDROR       ;PERFORM LOGIAL OR OF FAILING ADDRESS
024306  005100                          COM     R0              ;GET R0 READY FOR AND
024310  040037  001276                  BIC     R0,ADRAND               ;PERFORM LOGICAL AND
024314  005100                          COM     R0              ;PUT R0 BACK AS IT WAS
024316  105737  001103                  TSTB    $ERFLG          ;IS HIS THE FIRST ERROR
024322  001002                          BNE     2$              ;BRANCH IF NOT FIRST ERROR
024324  104201                          ERROR   201             ;NO REGISTER RESPONSE.
024326  000401                          BR      3$              ;BRANCH TO EXIT
024330  104301                  2$:     ERROR   301             ;CONTINUE NO RESPONSE TABLE
024332  012737  177777  024234  3$:     MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
024340  013746  001306                  MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
024344  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
024350  000006                          RTT                     ;RETURN TO THE TEST

                                .SBTTL  DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
                                ;;*************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
                                ;*      FOUND IN BOTH P.A.R.'S AND P.D.R.'S.  A 'LOGICAL OR'' AND A
                                ;*      'LOGICAL AND'' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
                                ;*      'ADDROR' AND 'ADRAND'.  THE LOG ON THE ADDITIONAL OR FAILING,
                                ;*      ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
```

```
                                ;*
                                ;;*****************************************************************************
024352                          DUALADR:                        ;STARTING LOCATION OF SUBROUTINE
024352  012637  001304                MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
024356  050037  001274                BIS     R0,ADDROR       ;LOGICAL OR OF WRITTEN ADDRESS
024362  005100                        COM     R0              ;GET R0 READY FOR AND
024364  040037  001276                BIC     R0,ADRAND               ;PERFORM LOGICAL AND
024370  005100                        COM     R0              ;PUT R0 BACK AS IT WAS
024372  050137  001264                BIS     R1,DATAOR       ;LOGICAL OR OF DUALED ADDRESS
024376  005101                        COM     R1              ;GET R1 READY FOR AND
024400  040137  001266                BIC     R1,DATAND               ;PERFORM LOGICAL AND
024404  005101                        COM     R1              ;PUT R1 BACK AS IT WAS
024406  105737  001103                TSTB    $ERFLG          ;SEE IF THIS IS FIRST ERROR
024412  001002                        BNE     1$              ;BRANCH IF NOT FIRST ERROR
024414  104202                        ERROR   202
024416  000401                        BR      2$              ;BRANCH TO EXIT
024420  104302                1$:     ERROR   302
024422  000177  154656        2$:     JMP     @OLDPC          ;RETURN TO TEST


                                .SBTTL  COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
                                ;;*****************************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
                                ;*      ERRORS OCCURRING WHEN TESTING THE P.A.R.'S AND P.D.R.'S.
                                ;*      THE 'LOGICAL OR' AND 'LOGICAL AND' OF VARIOUS DATA WILL BE
                                ;*      MAINTAINED AS FOLLOWS:
                                ;*      ADDRESSES OF FAILING REGISTERS IN 'ADDROR' AND 'ADRAND'
                                ;*      DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
                                ;*      PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
                                ;*
                                ;;*****************************************************************************
024426                          PARCOUNT:                       ;STARTING ADDRESS OF THIS SUBROUTINE
024426  012637  001304                MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
024432  050037  001274                BIS     R0,ADDROR       ;LOGICAL OR OF FAILING ADDRESS
024436  005100                        COM     R0              ;GET R0 READY FOR AND
024440  040037  001276                BIC     R0,ADRAND               ;PERFORM LOGICAL AND
024444  005100                        COM     R0              ;PUT R0 BACK AS IT WAS
024446  050137  001270                BIS     R1,PATTOR       ;LOGICAL OR OF PATTERN LOADED
024452  005101                        COM     R1              ;GET R1 READY FOR AND
024454  040137  001272                BIC     R1,PATAND               ;PERFORM LOGICAL AND
024460  005101                        COM     R1              ;PUT R1 BACK AS IT WAS
024462  050237  001264                BIS     R2,DATAOR       ;LOGICAL OR OF DATA FETCHED
024466  005102                        COM     R2              ;GET R2 READY FOR AND
024470  040237  001266                BIC     R2,DATAND               ;PERFORM LOGICAL AND
024474  005102                        COM     R2              ;PUT R2 BACK AS IT WAS
024476  105737  001103                TSTB    $ERFLG          ;SEE IF THIS IS THE FIRST ERROR
024502  001002                        BNE     1$              ;BRANCH IF NOT FIRST ERROR
024504  104203                        ERROR   203
024506  000401                        BR      2$              ;BRANCH TO EXIT
024510  104303                1$:     ERROR   303
024512  000177  154566        2$:     JMP     @OLDPC          ;RETURN TO TEST


                                .SBTTL  ********************* TRAP HANDLING ROUTINES *********************
```

```
                                        .SBTTL  CPU TRAP HANDLER ROUTINE
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
                                        ;*      'ERRVEC' (000004).  IF THIS SUBROUTINE IS ENTERED BY A SECOND
                                        ;*      TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
                                        ;*      IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
                                        ;*      UNEXPECTED ERROR MESSAGE IS GIVEN.  IF THE WORD 'CPUEXP' IS
                                        ;*      NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
                                        ;*      'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED.  'PCPUER' CAN
                                        ;*      BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
                                        ;*      IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
                                        ;*
                                        ;;*************************************************************************
024516  005227          CPUER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
024520  177777          CPFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
024522  001401                  BEQ     10$             ;BRANCH IF FIRST TIME IN
024524  000000                  HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                        ;I FINISHED REPORTING THE FIRST ERROR
                                                        ;THE SECOND ENTRY ADDRESS IS ON THE
                                                        ;STACK AND THE FIRST ERROR CONDITION
                                                        ;IS PROBABLY STILL LOCKED UP .
024526  012637  001304  10$:     MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
024532  012637  001306          MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
024536  013737  177766  001256  MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
024544  013737  001304  001260  MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
024552  005737  001222          TST     CPUEXP          ;SEE IF ANY CONDITION WAS EXPECTED
024556  001406                  BEQ     2$              ;BRANCH IF NO TRAP WAS EXPECTED
024560  023737  001256  001222  CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED ERROR OCCURED
024566  001403                  BEQ     1$              ;BRANCH IF ERROR CODES MATCH
024570  104001                  ERROR   1               ;ERROR TYPE OUT ITEM 1
024572  000401                  BR      1$              ;SKIP NEXT INSTRUCTION
024574  104002          2$:     ERROR   2               ;ERROR ITEM 2 NO CPU TRAP EXPECTED
024576  005037  177766  1$:     CLR     CPUERR          ;CLEAR CPU ERROR REGISTER
024602  012737  177777  024520  MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
024610  013746  001306          MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
024614  013746  001304          MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
024620  000006                  RTT                     ;RETURN FROM INTERRUPT OR ABORT


                                        .SBTTL  CACHE TRAPS AND ABORTS HANDLER ROUTINE
                                        ;;*************************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS.  IF
                                        ;*      THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
                                        ;*      BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED.  ON THE
                                        ;*      SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
                                        ;*      AFTER THE ABORT IS REPORTED.
                                        ;*      IF THE PARITY ERROR IS A TRAP THE CACHE WILL BE CLEANED UP BY
                                        ;*      REMOVING THE BAD WORD THAT MAY BE IN THE CACHE, AND THE TEST
                                        ;*      WILL BE CONTINUED AFTER THE PARITY ERROR IS REPORTED.
                                        ;*
                                        ;;*************************************************************************
024622  005227          MEMER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
024624  177777          PAFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
```

```
4049  024626  001401                        BEQ     10$             ;BRANCH IF FIRST TIME IN
4050  024630  000000                        HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
4051                                                                  ;I FINISHED REPORTING THE FIRST ERROR
4052                                                                  ;THE SECOND ENTRY ADDRESS IS ON THE
4053                                                                  ;STACK AND THE FIRST ERROR CONDITION
4054                                                                  ;IS PROBABLY STILL LOCKED UP .
4055  024632  012737  025052  000114  10$:     MOV     #5$,CACHVEC     ;SET CACHE VECTOR TO RTI UNTIL THE
4056                                                                  ;THE CACHE HAS BEEN FIXED.
4057  024640  012637  001304                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
4058  024644  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
4059  024650  013737  177740  001232          MOV     @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG
4060  024656  013737  177742  001234          MOV     @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR
4061  024664  013737  177744  001236          MOV     @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER
4062  024672  013737  177746  001240          MOV     @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT
4063  024700  013737  177750  001242          MOV     @#MAINT,PMAINT  ;SAVE MAINTENENCE REGISTER
4064  024706  013737  177752  001244          MOV     @#HITMIS,PHITMI ;SAVE HIT/MISS REGISTER
4065  024714  013737  001304  001260          MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
4066  024722  032737  000014  001236          BIT     #14,PPARER      ;WAS THIS A MAIN MEMORY REFERENCE
4067  024730  001005                          BNE     1$              ;BRANCH IF IT WAS A MAIN MEMORY ERROR
4068  024732  012737  024622  000114          MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
4069  024740  104003                          ERROR   3               ;UNEXPECTED CACHE PARITY ERROR
4070  024742  000415                          BR      2$              ;BRANCH TO EXIT POINT
4071  024744  010046                  1$:     MOV     R0,-(KSP)       ;SAVE R0 ON STACK
4072  024746  013700  001232                  MOV     PLOADR,R0       ;PUT LOW 16 BITS OF BAD ADDR IN R0
4073  024752  042700  176000                  BIC     #176000,R0      ;CLEAR UPPER 6 BITS OF ADDRESS
4074  024756  074027  000002                  XOR     R0,#BIT1        ;REFERENCE OTHER WORD OF PAIR
4075  024762  005710                          TST     (R0)            ;READ AT SAME INDEX AS BAD WORD
4076  024764  012600                          MOV     (KSP)+,R0       ;RESTORE R0 FROM STACK
4077  024766  012737  024622  000114          MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
4078  024774  104004                          ERROR   4               ;UNEXPECTED MAIN MEMORY PARITY ERROR
4079  024776  005737  001312          2$:     TST     RETRY           ;ARE YOU RETRYING THIS TEST?
4080  025002  001006                          BNE     3$              ;BRANCH IF THIS IS SECOND TRY
4081  025004  005237  001312                  INC     RETRY           ;SET RETRY FLAG
4082  025010  013737  001106  001304          MOV     $LPADR,OLDPC    ;RETURN TO START OF THIS TEST
4083  025016  000403                          BR      4$              ;BRANCH TO EXIT
4084  025020  013737  001314  001304  3$:     MOV     NXTTST,OLDPC    ;RETURN TO START OF NEXT TEST
4085                                                                  ;SINCE THIS IS THE SECOND ABORT
4086  025026  013737  001236  177744  4$:     MOV     PPARER,MEMERR   ;CLEAR MEMORY ERROR REGISTER
4087  025034  012737  177777  024624          MOV     #-1,PAFLAG      ;RESTORE A NEGATIVE ONE FOR NEXT TIME
4088  025042  013746  001306                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
4089  025046  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
4090  025052  000006                  5$:     RTT                     ;RETURN FROM INTERRUPT.
4091
4092
4093
4094                                  .SBTTL  MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
4095                                  ;;**************************************************************************
4096                                  ;*
4097                                  ;*      THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS
4098                                  ;*      AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM.  ANY M.M.
4099                                  ;*      ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED
4100                                  ;*      AND WILL BE REPORTED AS SUCH.  THIS MAY OCCUR BECAUSE SOME LOGIC
4101                                  ;*      THAT IS TO INHIBIT A TRAP HAS FAILED.
4102                                  ;*      THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO
4103                                  ;*      OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE
4104                                  ;*      CONTENTS OF 'PMMR0' (SAME AS 'MMR0') AN ERROR IS REPORTED.
```

```
                                        ;*
                                        ;;*************************************************************************
025054  005227                  MMTRAP: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
025056  177777                  MMFLAG: .WORD   -1              ;FLAG SHOULD BE NEG ONE
025060  001401                          BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
025062  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE
                                                                ;STACK AND THE FIRST ERROR CONDITION
                                                                ;IS PROBABLY STILL LOCKED UP .
025064  011637  001260          10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
025070  012637  001304                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
025074  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
025100  013737  177572  001246          MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
025106  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE AUTO INC/DEC REGISTER
025114  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
025122  005737  001224                  TST     MMEXP           ;SEE IF ANY M.M. TRAPS WERE EXPECTED
025126  001002                          BNE     5$              ;BRANCH IF ONE WAS EXPECTED
025130  104005                          ERROR   5               ;UNEXPECTED M.M. ABORT OR TRAP
025132  000405                          BR      1$              ;BRANCH TO EXIT
025134  023737  001224  001246  5$:     CMP     MMEXP,PMMR0     ;SEE IF ABORT OR TRAP WAS EXPECTED
025142  001401                          BEQ     1$              ;BRANCH IF CONDITION CORRECT
025144  104006                          ERROR   6               ;ERROR TYPE OUT ITEM 6.
025146  042737  177376  177572  1$:     BIC     #177376,@#MMR0  ;CLEAR ALL BITS EXCEPT 0 AND 8
025154  012737  177777  025056          MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
025162  013746  001306                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
025166  013746  001304                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
025172  000006                          RTT                     ;RETURN TO MAIN PROGRAM


                                .SBTTL  D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE
                                ;;*************************************************************************
                                ;*
                                ;*      THIS ROUTINE WILL BE ENTERED IF A MEMORY MANAGEMENT ABORT
                                ;*      OCCURS DURING THE D-SPACE ENABLE TESTS.  IF THE ABORT IS A
                                ;*      NON-RESIDENT ABORT THE PROBLEM IS PROBABLY IN THE D-SPACE
                                ;*      ENABLE LOGIC ON PAGES 'SAPK' AND 'SSRB' IN THE PRINTS.
                                ;*      IN ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 2 & 3
                                ;*      ARE MAPPED NON-RESIDENT AND I-SPACE PAGE 4 IS MAPPED NON-
                                ;*      RESIDENT.  ALL OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE.
                                ;*      THEREFORE IF THE N.R. PAGE IS 2 OR 3 YOU ARE PROBABLY NOT
                                ;*      FORCING I-SPACE WHEN YOU SHOULD.  BUT IF THE N.R. PAGE IS 4
                                ;*      YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD ALLOW D-SPACE
                                ;*
                                ;;*************************************************************************
025174                  NODSPAC:                                ;STARTING ADDRESS FOR ABORT SERVICE
025174  005227                  INC     (PC)+                   ;MAKE FLAG ZERO IF FIRST TIME
025176  177777          NDFLAG: .WORD   -1                      ;FLAG SHOULD BE NEG ONE
025200  001401                  BEQ     10$                     ;BRANCH IF FIRST TIME INTO ROUTINE
025202  000000                  HALT                            ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                ;I FINISHED REPORTING THE FIRST ERROR
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE
                                                                ;STACK AND THE FIRST ERROR CONDITION
                                                                ;IS PROBABLY STILL LOCKED UP .
025204  011637  001260          10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
025210  012637  001304                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
```

```
                                                MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
                                                MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
                                                MOV     MMR1,PMMR1      ;SAVE AUTO INC/DEC REGISTER
                                                MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
                                                TST     PMMR0           ;WAS ABORT NON-RESIDENT?
                                                BPL     1$              ;BRANCH IF NOT, IT IS UNEXPECTED.
                                                ERROR   132             ;D-SPACE ENABLE FAULTY
                                                BR      2$              ;BRANCH TO EXIT
                                        1$:     ERROR   5               ;UNEXPECTED M.M. ABORT
                                        2$:     BIC     #177376,MMR0    ;CLEAR ALL BITS EXCEPT 0 AND 8
                                                MOV     #-1,NDFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
                                                MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
                                                MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
                                                RTT                     ;RETURN TO MAIN PROGRAM


                                        .SBTTL  TRAP ROUTINES FOR ABORT IN SUPERVISOR OR USER MODE
                                        ;;**************************************************************************
                                        ;*
                                        ;*      THESE NEXT THREE SUBROUTINES ARE USED FOR THE TESTS THAT VERIFY
                                        ;*      THE VECTOR IS PICKED UP FROM KERNEL SPACE DURING AN ABORT.
                                        ;*      'KERVEC' IS WHERE I SHOULD GO SINCE IT IS AT 250 WHICH IS
                                        ;*              KERNEL SPACE VIRTUAL 250.
                                        ;*      'SUPVEC' IS AT 350 WHICH IS SUPERVISOR VIRTUAL 250
                                        ;*      'USEVEC' IS AT 450 WHICH IS USER SPACE VIRTUAL 250
                                        ;*
                                        ;*      THEY ALL READ THE PROCESSOR STATUS WHICH IS DIFFERENT FOR EACH
                                        ;*      VECTOR AND THEN JUMP TO 'RDMMR0'.  'RDMMR0' READS MMR0, MMR1,
                                        ;*      AND MMR2 AND RETURNS TO THE TEST WHERE THEY ARE TESTED.
                                        ;*
                                        ;;**************************************************************************
                                        KERVEC: MOV     PSW,R0          ;PUT PROCESSOR STATUS INTO R0
                                                MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
                                                MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
                                                CMPB    #340,R0         ;SEE IF CORRECT PSW WAS PICKED UP
                                                BEQ     1$              ;BRANCH IF PSW IS CORRECT
                                                ERROR   122             ;WRONG PSW PICKED UP
                                        1$:     JMP     RDMMR0          ;JUMP TO READ MMR0

                                        SUPVEC: MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
                                                MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
                                                MOV     PSW,R0          ;READ PSW FOR ERROR PRINT OUT
                                                ERROR   123             ;WRONG VECTOR, POSSIBLE WRONG PSW
                                                JMP     RDMMR0          ;GO READ MMR0

                                        USEVEC: MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
                                                MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
                                                MOV     PSW,R0          ;READ PSW FOR ERROR PRINT OUT
                                                ERROR   124             ;WRONG VECTOR, POSSIBLE WRONG PSW


                                        RDMMR0: MOV     MMR0,PMMR0      ;READ MMR0 TO BE CHECKED LATER
                                                MOV     MMR1,PMMR1      ;READ MMR1 FOR POSSIBLE ERROR REPORT
                                                MOV     MMR2,PMMR2      ;READ MMR2 FOR POSSIBLE ERROR REPORT
                                                MOV     OLDPS,-(KSP)    ;PUSH OLD PROCESSOR STATUS ON STACK
                                                MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
```

```
 4217  025426  000006                RTT                     ;RETURN TO TEST TO CHECK PMMR0
 4218
 4219
```

```
                                        .SBTTL
                                        .SBTTL  ************ ENTRY POINT 1 --- STARTING ADDRESS 200 ************
                                        .SBTTL  ********** TEST CODE STARTS AT ADDRESS 40000 (PAGE 2) **********
                040000                  .=40000

040000  012737  000000  001220  STRT1:  MOV     #0,FSTTST       ;LOAD INDEX TO START TABLE
040006  000433                  STRT2:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040010  012737  000002  001220          MOV     #2,FSTTST       ;LOAD INDEX TO START TABLE
040016  000427                  STRT3:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040020  012737  000004  001220          MOV     #4,FSTTST       ;LOAD INDEX TO START TABLE
040026  000423                  STRT4:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040030  012737  000006  001220          MOV     #6,FSTTST       ;LOAD INDEX TO START TABLE
040036  000417                  STRT5:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040040  012737  000010  001220          MOV     #10,FSTTST      ;LOAD INDEX TO START TABLE
040046  000413                  STRT6:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040050  012737  000012  001220          MOV     #12,FSTTST      ;LOAD INDEX TO START TABLE
040056  000407                  STRT7:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040060  012737  000014  001220          MOV     #14,FSTTST      ;LOAD INDEX TO START TABLE
040066  000403                  STRT8:  BR      START           ;BRANCH TO STARTING ADDRESS OF PROGRAM
040070  012737  000016  001220          MOV     #16,FSTTST      ;LOAD INDEX TO START TABLE


040076  012706  001100          START:  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
040102  005026                  2$:     CLR     (R6)+           ;;CLEAR MEMORY LOCATION
040104  022706  001136                  CMP     #$TKS,R6        ;;DONE?
040110  001374                          BNE     2$              ;;LOOP BACK IF NO
040112  012706  001100                  MOV     #KERSTK,KSP     ;;SET UP THE KERNEL STACK POINTER
040116  012737  021312  000020          MOV     #$SCOPE,IOTVEC  ;;IOT VECTOR FOR SCOPE ROUTINE
040124  012737  000340  000022          MOV     #340,IOTVEC+2   ;;PRIORITY LEVEL 7
040132  012737  021576  000030          MOV     #$ERROR,EMTVEC  ;;EMT VECTOR FOR ERROR ROUTINE
040140  012737  000340  000032          MOV     #340,EMTVEC+2   ;;PRIORITY LEVEL 7
040146  012737  023530  000034          MOV     #$TRAP,TRAPVEC  ;;TRAP VECTOR FOR TRAP CALLS
040154  012737  000340  000036          MOV     #340,TRAPVEC+2  ;;PRIORITY LEVEL 7
040162  012737  023572  000024          MOV     #$PWRDN,PWRVEC  ;;POWER FAILURE VECTOR
040170  012737  000340  000026          MOV     #340,PWRVEC+2   ;;PRIORITY LEVEL 7
040176  013737  021056  021050          MOV     $ENDCT,$EOPCT   ;;SET UP END-OF-PROGRAM COUNTER
040204  005037  001204                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
040210  005037  001206                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
040214  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
040222  012737  021276  000014          MOV     #$RTRN,TBITVEC  ;;SET 'T' BIT VECTOR TO $RTRN
040230  012737  000340  000016          MOV     #340,TBITVEC+2  ;;PRIORITY LEVEL 7
040236  012737  000006  021276          MOV     #RTT,$RTRN      ;;SET $RTRN TO A RTT
040244  005037  021304                  CLR     $TBIT           ;;CLEAR 'T' BIT SWITCH
040250  012737  040250  001106          MOV     #.,$LPADR       ;;INITIALIZE LOOP ADDRESS
040256  012737  040256  001110          MOV     #.,$LPERR       ;;INITIALIZE LOOP ON ERROR

040264  005037  177572          LOOP:   CLR     MMR0            ;GET INTO 16 BIT MODE ON SECOND PASS
040270  005037  172516                  CLR     MMR3            ;TURN OFF EVERY THING POSSIBLE
040274  012700  177777                  MOV     #-1,R0          ;PUT NEG ONE IN R0 TO INITIALIZE FLAGS
040300  010037  024520                  MOV     R0,CPFLAG       ;INITIALIZE CPU ERROR FLAG
040304  010037  024624                  MOV     R0,PAFLAG       ;INITIALIZE PARITY ERROR FLAG
040310  010037  025056                  MOV     R0,MMFLAG       ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
040314  010037  025176                  MOV     R0,NDFLAG       ;INITIALIZE NO D-SPACE TRAP FLAG
040320  010037  177744                  MOV     R0,MEMERR       ;CLEAR PARITY ERROR REGISTER (177744)
```

```
 4276  040324  010037  177766                MOV     R0,CPUERR       ;CLEAR CPU ERROR REGISTER (177766)
 4277  040330  005037  001222                CLR     CPUEXP          ;NOT EXPECTING ANY CPU TRAPS
 4278  040334  005037  001224                CLR     MMEXP           ;NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
 4279  040340  012737  024516  000004        MOV     #CPUER,ERRVEC   ;LOAD ADDRESS OF CPU TRAP ROUTINE
 4280  040346  012737  000340  000006        MOV     #340,ERRVEC+2   ;SET PRIORITY LEVEL 7
 4281  040354  012737  024622  000114        MOV     #MEMER,CACHVEC  ;LOAD ADDRESS OF PARITY TRAP ROUTINE
 4282  040362  012737  000340  000116        MOV     #340,CACHVEC+2  ;SET PRIORITY LEVEL 7
 4283  040370  012737  025054  000250        MOV     #MMTRAP,MMVEC   ;LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
 4284  040376  012737  000340  000252        MOV     #340,MMVEC+2    ;SET PRIORITY LEVEL 7
 4285  040404  004737  024174                JSR     PC,CLEANUP      ;INITIALIZE ALL ERROR LOCATIONS
 4286  040410  012706  001100                MOV     #KERSTK,KSP     ;SET UP KERNEL STACK POINTER
 4287  040414  005227  177777                INC     #-1             ;;FIRST TIME?
 4288  040420  001024                        BNE     64$             ;;BRANCH IF NO
 4289  040422  022737  021230  000042        CMP     #$ENDAD,@#42    ;;ACT-11?
 4290  040430  001420                        BEQ     64$             ;;BRANCH IF YES
 4291  040432  104400  040440                TYPE    ,65$            ;;TYPE ASCIZ STRING
 4292  040436  000415                        BR      64$             ;;GET OVER THE ASCIZ
 4293                              ;;65$:    .ASCIZ  <CRLF>?CEKBEC-C 11/70 MEM MGMT?<CRLF>
 4294  040472                      64$:
 4295  040472  013700  001220                MOV     FSTTST,R0       ;LOAD START TABLE INDEX
 4296  040476  001417                        BEQ     TST1            ;;START WITH TEST ONE IF ZERO
 4297  040500  012737  040000  177776        MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
 4298  040506  012706  000700                MOV     #SUPSTK,SSP     ;SET UP SUPERVISOR STACK POINTER
 4299  040512  012737  140000  177776        MOV     #140000,PSW     ;GO TO USER MODE
 4300  040520  012706  000600                MOV     #USESTK,USP     ;SET UP USER STACK POINTER
 4301  040524  012737  000340  177776        MOV     #340,PSW        ;GO TO KERNEL MODE PRIORITY LEVEL 7
 4302  040532  000170  001334                JMP     @STRTAB(R0)     ;JUMP TO CORRECT ENTRY POINT
 4303
 4304
 4305                              ;*        THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
 4306                              ;*        LOGIC FOR THE INTERNAL REGISTERS ON PAGE 'SCCE' AND TO MAKE
 4307                              ;*        SOME DETERMINATION ABOUT THE RESPONDING ADDRESSES.  IF AN
 4308                              ;*        ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
 4309                              ;*        AN ERROR WILL BE FLAGGED.  THE MULTIPLEXERS AND INTERNAL BUS
 4310                              ;*        DRIVERS ON PAGES 'SCCM' AND 'SCCN' ARE ALSO UTILIZED IN
 4311                              ;*        THESE TESTS AND COULD BE THE SOURCE OF ANY PROBLEMS ENCOUNTERED.
 4312
 4313
 4314                              ;;*****************************************************************
 4315                              ;*TEST 1         TRY TO READ ALL CPU REGISTERS
 4316                              ;*
 4317                              ;*        THIS TEST ENSURES THAT SOMETHING RESPONDS TO ADDRESSES 177760
 4318                              ;*        THRU 177776.  IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED
 4319                              ;*        THAT A REGISTER HAS TIMED OUT, AND AN ERROR IS REPORTED.
 4320                              ;*
 4321                              ;*        ERRORS THAT OCCUR IN THIS TEST ARE REPORTED FROM AN AREA
 4322                              ;*        AT THE END OF THIS TEST, STARTING AT '50$'.
 4323                              ;*
 4324                              ;;*****************************************************************
 4325  040536                      TST1:
 4326  040536  012737  040574  001106        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
 4327  040544  012737  040574  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
 4328  040552  012737  000001  001102        MOV     #1,$TSTNM       ;LOAD TEST NUMBER INTO MEMORY
 4329  040560  013737  001102  177570        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
 4330  040566  012737  040660  001314        MOV     #TST2,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST
 4331                                                                ;FOR ESCAPE ON PARITY ERRORS
```

```
4332  040574  012737  040620  001110  20$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
4333  040602  012737  040650  000004          MOV     #50$,ERRVEC     ;SET TIME OUT VECTOR TO SPECIAL ROUTINE
4334  040610  012706  001100                  MOV     #1100,KSP       ;SET KERNEL STACK POINTER TO 1100
4335  040614  012700  177760                  MOV     #177760,R0      ;PUT FIRST CPU REGISTER ADDRESS IN R0
4336  040620  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4337  040622  011001                          MOV     (R0),R1         ;TRY TO READ THE CPU REGISTERS
4338  040624  062700  000002          100$:   ADD     #2,R0           ;POINT TO NEXT CPU REGISTER
4339  040630  001373                          BNE     1$              ;BRANCH IF NOT ALL READ YET
4340  040632  012737  040574  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4341  040640  012737  024516  000004          MOV     #CPUER,ERRVEC   ;SET UP C.P.U. ERROR VECTOR
4342  040646  000404                          BR      TST2            ;;TEST OVER BRANCH TO NEXT TEST
4343
4344
4345  040650  062706  000004          50$:    ADD     #4,KSP          ;RE ADJUST KERNEL STACK POINTER
4346  040654  104024                          ERROR   24              ;CPU REGISTER TIMED OUT
4347  040656  000762                          BR      100$            ;GO BACK AND FINISH TEST
4348
4349
4350                                  ;;*****************************************************************
4351                                  ;*TEST 2        SYSTEM SIZE REGISTERS
4352                                  ;*
4353                                  ;*      BOTH THE LO SIZE AND THE HI SIZE REGISTERS ARE READ TO SEE THAT
4354                                  ;*      THEY HOLD SOMETHING THAT LOOKS CORRECT.  (IE. LO SIZE SHOULD
4355                                  ;*      HAVE 377 IN ITS LOW BYTE, AND HI SIZE SHOULD BE ZERO)  THEY
4356                                  ;*      ARE ALSO VERIFIED TO BE READ ONLY REGISTERS.
4357                                  ;*
4358                                  ;;*****************************************************************
4359  040660                          TST2:
4360  040660  000004                          SCOPE
4361  040662  012737  041014  001314          MOV     #TST3,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
4362                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4363  040670  012737  040702  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
4364  040676  012700  177760                  MOV     #177760,R0      ;PUT ADDRESS OF SIZE LO REGISTER INTO R0
4365  040702  000240                  21$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4366  040704  011001                          MOV     (R0),R1         ;READ SYSTEM SIZE LO REGISTER
4367  040706  122701  000377                  CMPB    #377,R1         ;SEE IF LOW BYTE IS ALL ONES
4368  040712  001401                          BEQ     1$              ;BRANCH IF LOW BYTE IS ALL ONES
4369  040714  104025                          ERROR   25              ;LOW BYTE OF LO SIZE IS NOT -1
4370  040716  012737  040724  001110  1$:     MOV     #22$,$LPERR     ;SET LOOP ON ERROR POINTER TO 22$
4371  040724  000240                  22$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4372  040726  005010                          CLR     (R0)            ;TRY TO CLEAR SIZE LO REGISTER
4373  040730  011002                          MOV     (R0),R2         ;READ SIZE LO REGISTER INTO R2
4374  040732  001002                          BNE     2$              ;BRANCH IF IT IS NOT ZERO
4375  040734  104026                          ERROR   26              ;COULD WRITE SIZE LO REG
4376  040736  010110                          MOV     R1,(R0)         ;RESTORE DATA TO ADDRESS
4377  040740  012737  040752  001110  2$:     MOV     #23$,$LPERR     ;SET LOOP ON ERROR POINTER TO 23$
4378  040746  012700  177762                  MOV     #177762,R0      ;PUT ADDRESS OF SIZE HIGH REG INTO R0
4379  040752  000240                  23$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4380  040754  011001                          MOV     (R0),R1         ;READ SYSTEM SIZE HIGH REGISTER
4381  040756  001401                          BEQ     3$              ;BRANCH IF IT IS ZERO
4382  040760  104027                          ERROR   27              ;(177762) IS NOT ZERO
4383  040762  012737  040770  001110  3$:     MOV     #24$,$LPERR     ;SET LOOP ON ERROR POINTER TO 24$
4384  040770  000240                  24$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
4385  040772  012710  177777                  MOV     #-1,(R0)        ;TRY TO LOAD SIZE HIGH REGISTER
4386  040776  011002                          MOV     (R0),R2         ;READ SIZE HIGH REGISTER
4387  041000  001402                          BEQ     4$              ;BRANCH IF REGISTER IS STILL ZERO
```

```
                                        ERROR   26              ;COULD WRITE SIZE REGISTER
                                        MOV     R1,(R0)         ;RESTORE DATA TO ADDRESS
041006  012737  040670  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                        ;;**************************************************************
                        ;*TEST 3         CPU ERROR REGISTER
                        ;*
                        ;*      THE CPU ERROR REGISTER IS TESTED TO SEE THAT IT IS CLEAR
                        ;*      AFTER A NEGATIVE ONE HAS BEEN LOADED INTO IT.  THERE IS NO
                        ;*      WAY TO SET ANY BITS UNDER PROGRAM CONTROL.
                        ;*
                        ;;**************************************************************
041014                  TST3:
041014  000004                 SCOPE
041016  012737  041060  001314          MOV     #TST4,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
041024  012737  041036  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
041032  012700  177766                  MOV     #177766,R0      ;LOAD ADDRESS OF C.P.U. ERROR REG
041036  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041040  012710  177777                  MOV     #-1,(R0)        ;WRITE A NEGATIVE ONE INTO CPU ERROR REG
041044  011001                          MOV     (R0),R1         ;READ C.P.U. ERROR REGISTER
041046  001401                          BEQ     2$              ;CPU ERROR REGISTER SHOULD BE ZERO
041050  104030                          ERROR   30              ;C.P.U. ERROR REGISTER NOT ZERO
041052  012737  041024  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                        ;;**************************************************************
                        ;*TEST 4         MICRO PROGRAM BREAK REGISTER
                        ;*
                        ;*      THE MICRO BREAK REGISTER IS A LOW BYTE ONLY REGISTER, WITH
                        ;*      THE UPPER BYTE ALWAYS ZERO.  THIS TEST LOADS A FULL WORD INTO
                        ;*      ADDRESS 177770 AND CHECKS TO SEE THAT ONLY THE LOW BYTE IS
                        ;*      STORED IN THE REGISTER.
                        ;*
                        ;;**************************************************************
041060                  TST4:
041060  000004                 SCOPE
041062  012737  041134  001314          MOV     #TST5,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
041070  012737  041102  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
041076  012700  177770                  MOV     #177770,R0      ;LOAD ADDRESS OF MICRO BREAK REGISTER
041102  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041104  012710  154044                  MOV     #154044,(R0)    ;ONLY LOW BYTE SHOULD BE LOADED
041110  011001                          MOV     (R0),R1         ;READ MICRO BREAK REGISTER
041112  012702  000044                  MOV     #044,R2         ;LOAD EXPECTED DATA INTO R2
041116  020102                          CMP     R1,R2           ;DID YOU REFERENCE THE MICRO BREAK REG
041120  001401                          BEQ     2$              ;BRANCH IF DATA MATCHES
041122  104032                          ERROR   32              ;DIDN'T LOAD MICRO BREAK REGISTER
041124  005010                  2$:     CLR     (R0)            ;LEAVE MICRO BREAK REGISTER ZERO
041126  012737  041070  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                        ;;**************************************************************
                        ;*TEST 5         PROGRAM INTERRUPT REQUEST REGISTER
```

```
                                ;*
                                ;*      THE PROGRAM INTERRUPT REQUEST REGISTER'S UPPER BYTE IS LOADED
                                ;*      DIRECTLY AND THE LOWER BYTE IS AN ENCODE OF THE UPPER BYTE.
                                ;*      THIS TEST LOADS A FULL WORD INTO ADDRESS 177772 AND CHECKS
                                ;*      THAT THE UPPER BYTE IS LOADED DIRECTLY AND THE LOWER BYTE
                                ;*      IS ENCODED CORRECTLY.
                                ;*
                                ;;*********************************************************************
041134                          TST5:
041134  000004                          SCOPE
041136  012737  041212  001314          MOV     #TST6,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
041144  012737  041160  001110  20$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
041152  000237                          SPL     7               ;SET PRIORITY LEVEL AT SEVEN
041154  012700  177772                  MOV     #177772,R0      ;LOAD ADDRESS OF PROGRAM INTERRUPT REG
041160  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041162  012710  004777                  MOV     #004777,(R0)    ;ONLY UPPER BYTE (LESS BIT 8) IS LOADED
                                                                ;DIRECTLY, LOWER BYTE IS ENCODED
041166  011001                          MOV     (R0),R1         ;READ PROGRAM INTERRUPT REGISTER
041170  012702  004146                  MOV     #004146,R2      ;LOAD EXPECTED DATA
041174  020102                          CMP     R1,R2           ;DID YOU REFERENCE PROGRAM INTERRUPT REG
041176  001401                          BEQ     2$              ;BRANCH IF IT WAS THE PROG. INTERRUPT
041200  104033                          ERROR   33              ;NOT THE PROG. INT. REQUEST REGISTER
041202  005010                  2$:     CLR     (R0)            ;LEAVE THE P.I.R. REGISTER ZERO
041204  012737  041144  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                ;;*********************************************************************
                                ;*TEST 6         STACK LIMIT REGISTER
                                ;*
                                ;*      THE STACK LIMIT REGISTER IS A HIGH BYTE ONLY REGISTER, SO THIS
                                ;*      TEST TRIES TO LOAD BOTH BYTES OF ADDRESS 177774 AND CHECKS
                                ;*      THAT ONLY THE HIGH BYTE IS LOADED.  THE LOW BYTE WILL ALWAYS
                                ;*      BE READ AS ZERO.
                                ;*
                                ;;*********************************************************************
041212                          TST6:
041212  000004                          SCOPE
041214  012737  041266  001314          MOV     #TST7,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
041222  012737  041234  001110  20$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
041230  012700  177774                  MOV     #177774,R0      ;LOAD ADDRESS OF STACK LIMIT REGISTER
041234  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041236  012710  000777                  MOV     #777,(R0)       ;ONLY HIGH BYTE SHOULD BE LOADED
041242  011001                          MOV     (R0),R1         ;READ STACK LIMIT REGISTER
041244  005010                          CLR     (R0)            ;LEAVE STACK LIMIT REGISTER CLEAR
041246  012702  000400                  MOV     #400,R2         ;LOAD EXPECTED DATA INTO R2
041252  020102                          CMP     R1,R2           ;DID YOU REFERENCE THE STACK LIMIT REG
041254  001401                          BEQ     2$              ;BRANCH IF IT WAS STACK LIMIT
041256  104034                          ERROR   34              ;NOT STACK LIMIT REGISTER
041260  012737  041222  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                ;;*********************************************************************
                                ;*TEST 7         PROCESSOR STATUS WORD
```

```
                                        ;*
                                        ;*      THIS TEST SETS THE PRIORITY LEVEL TO 7 AND CLEARS THE
                                        ;*      CONDITION CODES AND CHECKS TO SEE THAT ADDRESS 177776
                                        ;*      HAS 340 IN ITS LOW BYTE.
                                        ;*
                                        ;;*****************************************************************
041266                                  TST7:
041266  000004                                SCOPE
041270  012737  041344  001314                MOV     #TST10,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
041276  012737  041314  001110          20$:  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
041304  012700  177776                      MOV     #177776,R0      ;LOAD ADDRESS OF P.S.W. INTO R0
041310  012702  177740                      MOV     #177740,R2      ;LOAD EXPECTED DATA INTO R2
041314  000237                        1$:   SPL     7               ;SET PRIORITY TO LEVEL SEVEN
041316  000257                              CCC                     ;CLEAR ALL CONDITION CODES
041320  000240                              NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041322  111001                              MOVB    (R0),R1         ;LOAD LOWER BYTE OF P.S.W. INTO R1
041324  042701  000020                      BIC     #TBIT,R1        ;CLEAR T-BIT IF IT IS ON
041330  020201                              CMP     R2,R1           ;SIGN EXTEND OF LOWER BYTE OF PSW
                                                                        ;LOWER BYTE SHOULD BE 340
041332  001401                              BEQ     2$              ;BRANCH IF PSW IS CORRECT
041334  104031                              ERROR   31              ;MUST HAVE READ SOME OTHER REGISTER
041336  012737  041276  001110          2$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                        ;;*****************************************************************
                                        ;*TEST 10       SET UP THE STACK POINTERS FOR REST OF TESTS
                                        ;*
                                        ;*      THIS TEST IS USED TO SET THE KERNEL STACK POINTER AT
                                        ;*      1100, THE SUPERVISOR STACK POINTER AT 700, AND THE USER
                                        ;*      STACK POINTER AT 600.  IT THEN RETURNS TO KERNEL MODE AND
                                        ;*      VERIFIES THAT THE KERNEL STACK POINTER IS STILL AT 1100.
                                        ;*
                                        ;;*****************************************************************
041344                                  TST10:
041344  000004                                SCOPE
041346  012737  041426  001314                MOV     #TST11,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
041354  005037  177776                      CLR     PSW             ;GET TO KERNEL MODE, PRIORITY 0
041360  012706  001100                      MOV     #KERSTK,KSP     ;SETUP KERNEL STACK POINTER
041364  012737  040000  177776              MOV     #40000,PSW      ;GET INTO SUPERVISOR MODE
041372  012706  000700                      MOV     #SUPSTK,SSP     ;SET UP SUPERVISOR STACK POINTER
041376  012737  140000  177776              MOV     #140000,PSW     ;GET INTO USER MODE
041404  012706  000600                      MOV     #USESTK,USP     ;SET UP USER STACK POINTER
041410  005037  177776                      CLR     PSW             ;GET BACK INTO KERNEL MODE
041414  010600                              MOV     KSP,R0          ;READ KSRNEL STACK POINTER
041416  022700  001100                      CMP     #KERSTK,R0      ;SEE IF KERNEL STACK IS STILL 1100
041422  001401                              BEQ     TST11           ;;BRANCH IF KERNEL STACK IS OKAY
041424  104035                              ERROR   35              ;KERNEL STACK POINTER IS NOT RIGHT


                                        .SBTTL  ************ ENTRY POINT 2 --- STARTING ADDRESS 204 ************
                                        .SBTTL  ** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **
                                        ;*
                                        ;*      THIS GROUP OF TESTS EXERCISES ALL OF THE READ/WRITE BITS
                                        ;*      IN MMR0, TESTS THE PROPER FUNCTIONING OF MMR1, AND TRIES A
```

```
                                ;*      FEW VIRTUAL ADDRESSES IN MMR2.  IT ALSO EXERCISES THE BITS
                                ;*      IN MMR3.  THESE TESTS SHOW THAT ONCE THE 'SSRC NO ERROR' FLIP-
                                ;*      FLOP IS SET MMR1 AND MMR2 STOP TRACKING THE C.P.U. OPERATIONS.
                                ;*


                                ;;*********************************************************************
                                ;*TEST 11       BIT TEST OF MEMORY MANAGEMENT REGISTER 0
                                ;*
                                ;*      THIS TEST TRIES TO SET AND CLEAR BITS <15:12> AND <09> OF
                                ;*      MMR0.  'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.
                                ;*      THE OTHER BITS OF MMR0 ARE CLOCKED ONLY ON MEMORY
                                ;*      MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
                                ;*      BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
                                ;*      RELOCATION ON THE DESTINATION CYCLE ONLY.  THESE BITS WILL
                                ;*      BE TESTED IN A LATER TEST.
                                ;*
                                ;;*********************************************************************
041426                          TST11:
                                        SCOPE
041426  000004
041430  012737  041632  001314        MOV     #TST12,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
041436                          ENTPT2:
041436  012737  041466  001106        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
041444  012737  041466  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
041452  012737  000011  001102        MOV     #11,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
041460  013737  001102  177570        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
041466  012700  177572          20$:  MOV     #MMR0,R0        ;PUT ADDRESS OF MMR0 IN R0
041472  012710  171000                MOV     #171000,(R0)    ;LOAD WRITABLE BITS IN MMR0
041476  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
041500  000005                        RESET                   ;'INIT', SHOULD CLEAR ALL BITS OF MMR0
041502  011001                        MOV     (R0),R1         ;READ MMR0
041504  001401                        BEQ     1$              ;BRANCH IF MMR0 IS CLEAR
041506  104007                        ERROR   7               ;MMR0 WON'T CLEAR
041510  012737  041516  001110  1$:   MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
041516  000240                11$:  NOP                     ;THIS IS SYNC POINT FOR SCOPING
041520  012710  171000                MOV     #171000,(R0)    ;SET BITS <15:12><9> OF MMR0
041524  011001                        MOV     (R0),R1         ;READ MMR0
041526  022701  171000                CMP     #171000,R1      ;SEE IF BITS <15:12><9> ARE SET
041532  001401                        BEQ     2$              ;BRANCH IF CORRECT BITS ARE SET
041534  104010                        ERROR   10              ;CAN'T SET 171000 IN MMR0
041536  012737  041544  001110  2$:   MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
041544  000240                12$:  NOP                     ;THIS IS SYNC POINT FOR SCOPING
041546  005010                        CLR     (R0)            ;CLEAR UPPER 7 BITS OF MMR0
041550  011001                        MOV     (R0),R1         ;READ MMR0
041552  001401                        BEQ     3$              ;BRANCH IF MMR0 IS CLEAR
041554  104007                        ERROR   7               ;MMR0 WON'T CLEAR
041556  012737  041574  001110  3$:   MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
041564  012702  100000                MOV     #BIT15,R2       ;SET BIT IN R2 TO FLOAT THRU MMR0
041570  012703  000007                MOV     #7,R3           ;PUT LOOP COUNT IN R3
041574  000240                5$:   NOP                     ;THIS A IS SYNC POINT FOR SCOPING
041576  010210                        MOV     R2,(R0)         ;LOAD R2 INTO MMR0
041600  011001                        MOV     (R0),R1         ;READ MMR0 IN R1
041602  005010                        CLR     (R0)            ;CLEAR MMR0
041604  010204                        MOV     R2,R4           ;PUT PATTERN IN R4
041606  042704  006777                BIC     #006777,R4      ;MASK OFF BITS <11:10><8:0>
```

```
4612  041612  020401                          CMP     R4,R1           ;COMPARE PATTERN WITH MMR0
4613  041614  001401                          BEQ     4$              ;BRANCH IF DATA MATCHES
4614  041616  104011                          ERROR   11              ;GOT WRONG DATA FROM MMR0
4615  041620  006002                  4$:     ROR     R2              ;SHIFT BIT TO RIGHT
4616  041622  077314                          SOB     R3,5$           ;LOOP BACK 6 TIMES
4617  041624  012737  041466  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4618
4619
4620                                  ;;*********************************************************************
4621                                  ;*TEST 12       BIT TEST OF MEMORY MANAGEMENT REGISTER 1
4622                                  ;*
4623                                  ;*      THIS TEST WILL CAUSE BITS IN MMR1 TO BE LOCKED BY SETTING
4624                                  ;*      BITS <15:13> IN MMR0.  THE REGISTER ONLY GETS CLOCKED ON
4625                                  ;*      AN INSTRUCTION THAT AUTO INCREMENTS OR AUTO DECREMENTS A
4626                                  ;*      REGISTER.  THE LOWER BYTE IS ALWAYS CLOCKED FIRST AND THE
4627                                  ;*      UPPER BYTE IS CLOCKED ONLY IF BOTH SOURCE AND DESTINATION
4628                                  ;*      AUTO INCREMENT OR DECREMENT A REGISTER.
4629                                  ;*      ALL COUNTS (+1,-1,+2,-2) THAT CAN BE GENERATED WITH JUST THE
4630                                  ;*      C.P. ARE CLOCKED INTO BOTH HIGH AND LOW BYTES.  ALL REGISTER
4631                                  ;*      NUMBERS ARE GENERATED, INCLUDING ALL 3 R6'S, AND ALL THE
4632                                  ;*      BITS THAT HOLD THE REGISTER NUMBER IN BOTH BYTES ARE TESTED.
4633                                  ;*      HOWEVER NOT ALL REGISTER NUMBERS ARE CLOCKED INTO BOTH HIGH
4634                                  ;*      AND LOW BYTES, BUT ENOUGH ARE USED TO TEST THE LOGIC.
4635                                  ;*
4636                                  ;;*********************************************************************
4637  041632                          TST12:
4638  041632  000004                          SCOPE
4639  041634  012737  042264  001314          MOV     #TST13,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4640                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4641  041642  012701  177574          20$:    MOV     #MMR1,R1        ;PUT ADDRESS OF MMR1 IN R1
4642  041646  012704  020000                  MOV     #BIT13,R4       ;SET READ ONLY BIT IN R4
4643  041652  012737  041660  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4644  041660  000240                  11$:    NOP                     ;THIS IS SYNC POINT FOR SCOPING
4645  041662  012737  100000  177572          MOV     #BIT15,@#MMR0   ;LOCK UP MMR1 LOWER BYTE 027
4646                                                                  ;UPPER BYTE 027-WORD 013427
4647  041670  012703  013427                  MOV     #013427,R3      ;PUT EXPECTED DATA IN R3
4648  041674  011102                          MOV     (R1),R2         ;READ MMR1 INTO R2
4649  041676  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4650  041700  001401                          BEQ     10$             ;BRANCH IF DATA MATCHES
4651  041702  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
4652  041704  012737  041712  001110  10$:    MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
4653  041712  000240                  12$:    NOP                     ;THIS IS SYNC POINT FOR SCOPING
4654  041714  000005                          RESET                   ;ISSUE INIT
4655  041716  011102                          MOV     (R1),R2         ;SEE IF MMR1 IS CLEARED
4656  041720  001401                          BEQ     1$              ;BRANCH IF MMR1 IS ZERO
4657  041722  104012                          ERROR   12              ;CAN'T CLEAR MMR1
4658  041724  012737  041732  001110  1$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4659  041732  012700  177572          13$:    MOV     #MMR0,R0        ;PUT ADDRESS OF MMR0 INTO R0
4660  041736  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
4661  041740  012720  040000                  MOV     #BIT14,(R0)+    ;LOCK UP MMR1-LOWER BYTE 027
4662                                                                  ;UPPER BYTE 020-WORD 010027
4663  041744  011102                          MOV     (R1),R2         ;READ MMR1
4664  041746  012703  010027                  MOV     #010027,R3      ;PUT EXPECTED DATA IN R3
4665  041752  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4666  041754  001401                          BEQ     2$              ;BRANCH IF DATA MATCHES
4667  041756  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
```

```
4668  041760  005037  177572          2$:     CLR     @#MMR0          ;CLEAR MMR0
4669  041764  012737  041772  001110          MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
4670  041772  012700  177574          14$:    MOV     #MMR0+2,R0      ;PUT ADDRESS OF MMR0 PLUS 2 IN R0
4671  041776  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
4672  042000  010440                          MOV     R4,-(R0)        ;LOCK UP MMR1-LOWER BYTE 360
4673                                                                  ;UPPER BYTE 000-WORD 000360
4674  042002  011102                          MOV     (R1),R2         ;READ MMR1
4675  042004  012703  000360                  MOV     #000360,R3      ;PUT EXPECTED DATA IN R3
4676  042010  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4677  042012  001401                          BEQ     3$              ;BRANCH IF DATA MATCHES
4678  042014  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
4679  042016  005010                  3$:     CLR     (R0)            ;CLEAR MMR0
4680  042020  012737  042026  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
4681  042026  012705  177573          15$:    MOV     #MMR0+1,R5      ;PUT ADDRESS OF MMR0'S UPPER BYTE IN R5
4682  042032  012702  001317                  MOV     #READON+1,R2    ;PUT ADDRESS OF READ ONLY BIT <13> IN R2
4683  042036  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
4684  042040  112225                          MOVB    (R2)+,(R5)+     ;LOCK UP MMR1-LOWER BYTE 012
4685                                                                  ;UPPER BYTE 015-WORD 006412
4686  042042  011102                          MOV     (R1),R2         ;READ MMR1
4687  042044  012703  006412                  MOV     #006412,R3      ;PUT EXPECTED DATA IN R3
4688  042050  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4689  042052  001401                          BEQ     4$              ;BRANCH IF DATA MATCHES
4690  042054  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
4691  042056  005010                  4$:     CLR     (R0)            ;CLEAR MMR0
4692  042060  012737  042066  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
4693  042066  012702  177574          16$:    MOV     #MMR0+2,R2      ;PUT ADDRESS OF MMR0'S UPPER BYTE
4694                                                                  ;PLUS 1 INTO R2
4695  042072  012705  001320                  MOV     #READON+2,R5    ;PUT ADDRESS OF READ ONLY BIT <13>
4696                                                                  ;PLUS 2 IN R5
4697  042076  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
```

```
4698  042100  114542                          MOVB    -(R5),-(R2)     ;LOCK UP MMR1-LOWER BYTE 375
4699                                                                    ;UPPER BYTE 372-WORD 175375
4700  042102  011102                          MOV     (R1),R2         ;READ MMR1
4701  042104  012703  175375                  MOV     #175375,R3      ;PUT EXPECTED DATA IN R3
4702  042110  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4703  042112  001401                          BEQ     5$              ;BRANCH IF DATA MATCHES
4704  042114  104013                          ERROR   13              ;MMR1 DID NOT TEACH PROPERLY
4705  042116  005010                  5$:     CLR     (R0)            ;CLEAR MMR0
4706  042120  012737  042140  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
4707  042126  012737  040000  177776          MOV     #40000,@#PSW    ;SET SUPERVISOR MODE
4708  042134  010637  001170                  MOV     SSP,@#$TMP0     ;SAVE SUPERVISOR STACK POINTER
4709  042140  012706  177574          17$:    MOV     #MMR0+2,SSP     ;PUT ADDRESS OF MMR0+2 IN SSP
4710  042144  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
4711  042146  012746  040000                  MOV     #40000,-(SSP)   ;LOCK UP MMR1-LOWER BYTE 027
4712                                                                    ;UPPER BYTE 366-WORD 173027
4713  042152  011102                          MOV     (R1),R2         ;READ MMR1
4714  042154  012703  173027                  MOV     #173027,R3      ;PUT EXPECTED DATA IN R3
4715  042160  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4716  042162  001401                          BEQ     6$              ;BRANCH IF DATA MATCHES
4717  042164  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
4718  042166  013706  001170          6$:     MOV     $TMP0,SSP       ;RESTORE THE SUPERVISOR STACK POINTER
4719  042172  005010                          CLR     (R0)            ;CLEAR MMR0
4720  042174  012737  042214  001110          MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
4721  042202  012737  140000  177776          MOV     #140000,@#PSW   ;SET USER MODE
4722  042210  010637  001170                  MOV     USP,@#$TMP0     ;SAVE USER STACK POINTER
4723  042214  012706  177572          18$:    MOV     #MMR0,USP       ;PUT ADDRESS OF MMR0 IN USP
4724  042220  000240                          NOP                     ;THIS IS SYNC POINT FOR SCOPING
4725  042222  012726  100000                  MOV     #100000,(USP)+  ;LOCK UP MMR1-LOWER BYTE 027
4726                                                                    ;UPPER BYTE 026-WORD 013027
4727  042226  011102                          MOV     (R1),R2         ;READ MMR1
4728  042230  012703  013027                  MOV     #013027,R3      ;PUT EXPECTED DATA IN R3
4729  042234  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
4730  042236  001401                          BEQ     7$              ;BRANCH IF DATA IS GOOD
4731  042240  104013                          ERROR   13              ;MMR1 DID NOT TRACK PROPERLY
4732  042242  013706  001170          7$:     MOV     @#$TMP0,USP     ;RESTORE USER STACK POINTER
4733  042246  005037  177776                  CLR     @#PSW           ;GO BACK TO KERNEL MODE
4734  042252  005037  177572                  CLR     @#MMR0          ;LET ALL M.M.R'S TRACK
4735  042256  012737  041642  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
4736
4737
4738                                  ;;*******************************************************************
4739                                  ;*TEST 13       BIT TEST OF MEMORY MANAGEMENT REGISTER 2
4740                                  ;*
4741                                  ;*      HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO
4742                                  ;*      SEE THAT IT IS TRACKING PROPERLY.  THEN IT WILL BE
4743                                  ;*      LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER
4744                                  ;*      IT IS ONCE LOCKED UP.  NOT ALL BITS WILL BE TESTED IN THIS
4745                                  ;*      TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2.  THIS
4746                                  ;*      CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS
4747                                  ;*      FUNCTIONING PROPERLY.
4748                                  ;*
4749                                  ;;*******************************************************************
4750  042264                          TST13:
4751  042264  000004                          SCOPE
4752  042266  012737  042440  001314          MOV     #TST14,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4753                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
```

```
4754  042274  012700  177572          20$:    MOV     #MMR0,R0        ;PUT ADDRESS OF MMR0 IN R0
4755  042300  012701  177576                  MOV     #MMR2,R1        ;PUT ADDRESS OF MMR2 IN R1
4756  042304  012737  042316  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4757  042312  005037  177572                  CLR     @#MMR0          ;MAKE SURE THAT MMR0 IS CLEAR
4758  042316  000240                  11$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4759  042320  011102                  21$:    MOV     (R1),R2         ;READ MMR2 TO R2
4760  042322  012703  042320                  MOV     #21$,R3         ;PUT ADDRESS OF LAST INST IN R3
4761  042326  020203                          CMP     R2,R3           ;SEE IF MMR2 HELD CORRECT ADDRESS
4762  042330  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
4763  042332  104014                          ERROR   14              ;MMR2 DID NOT TRACK PROPERLY
4764  042334  012737  042342  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
4765  042342  000240                  12$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4766  042344  013702  177576          22$:    MOV     @#MMR2,R2       ;READ MMR2 TO R2
4767  042350  012703  042344                  MOV     #22$,R3         ;PUT ADDRESS OF LAST INST IN R3
4768  042354  020203                          CMP     R2,R3           ;SEE IF MMR2 HELD CORRECT ADDRESS
4769  042356  001401                          BEQ     2$              ;BRANCH IF DATA MATCHES
4770  042360  104014                          ERROR   14              ;MMR2 DID NOT TRACK PROPERLY
4771  042362  012737  042370  001110  2$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4772  042370  000240                  13$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4773  042372  012710  040000          23$:    MOV     #40000,(R0)     ;LOCK UP MMR1 AND MMR2
4774  042376  011102                          MOV     (R1),R2         ;READ MMR2 INTO R2
4775  042400  012703  042372                  MOV     #23$,R3         ;PUT LOCKING INST'S ADDRESS IN R3
4776  042404  020203                          CMP     R2,R3           ;SEE IF MMR2 HOLDS RIGHT ADDRESS
4777  042406  001401                          BEQ     3$              ;BRANCH IF DATA MATCHES
4778  042410  104014                          ERROR   14              ;MMR2 DID NOT TRACK PROPERLY
4779  042412  012737  042420  001110  3$:     MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
4780  042420  000005                  14$:    RESET                   ;ISSUE INIT TO CLEAR MMR1 & 2
4781  042422  000240                  15$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4782  042424  011102                  25$:    MOV     (R1),R2         ;READ MMR2 INTO R2
4783  042426  012703  042424                  MOV     #25$,R3         ;PUT ADDRESS OF LAST INST IN R3
4784  042432  020203                          CMP     R2,R3           ;SEE IF MMR2 IS STILL TRACKING
4785  042434  001401                          BEQ     TST14           ;;GO TO NEXT TEST IF IT IS
4786  042436  104014                          ERROR   14              ;MMR2 DID NOT TRACK PROPERLY
4787
4788                                  ;;*****************************************************************
4789                                  ;*TEST 14       BIT TEST OF MEMORY MANAGEMENT REGISTER 3
4790                                  ;*
4791                                  ;*      THIS TEST SETS AND CLEARS BITS <05:04> AND <02:00> OF MMR3
4792                                  ;*      IT DOES NOT TEST THAT THE BITS FUNCTION PROPERLY SINCE THAT
4793                                  ;*      REQUIRES MORE LOGIC.  BUT IT DOES TEST THE REGISTER AND THE
4794                                  ;*      DATA PATH TO  AND FROM THE REGISTER.  THE PROPER FUNCTIONING
4795                                  ;*      OF THESE BITS IS TESTED LATER IN SEVERAL TESTS.
4796                                  ;*
4797                                  ;;*****************************************************************
4798  042440                          TST14:
4799  042440  000004                          SCOPE
4800  042442  012737  042604  001314          MOV     #TST15,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
4801                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
4802  042450  012700  172516          20$:    MOV     #MMR3,R0        ;PUT ADDRESS OF MMR3 IN R0
4803  042454  012701  000001                  MOV     #1,R1           ;SET BIT TO FLOAT THRU MMR3
4804  042460  005010                          CLR     (R0)            ;CLEAR MMR3
4805  042462  011002                          MOV     (R0),R2         ;READ MMR3 INTO R2
4806  042464  001401                          BEQ     5$              ;BRANCH IF ZERO
4807  042466  104015                          ERROR   15              ;CAN'T CLEAR MMR3
4808  042470  012703  000006          5$:     MOV     #6,R3           ;SET UP LOOP COUNT
4809  042474  012737  042502  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
```

```
4810  042502  000240                  1$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4811  042504  010110                          MOV     R1,(R0)         ;LOAD MMR3
4812  042506  011002                          MOV     (R0),R2         ;READ MMR3 INTO R2
4813  042510  020102                          CMP     R1,R2           ;DOES DATTA MATHC PATTERN
4814  042512  001404                          BEQ     2$              ;BRANCH IF IT MATCHES
4815  042514  032701  000010                  BIT     #BIT3,R1        ;WAS THIS BIT 3
4816  042520  001001                          BNE     2$              ;BRANCH IF IT WAS
4817  042522  104016                          ERROR   16              ;MMR3 HELD WRONG DATA
4818  042524  006301                  2$:     ASL     R1              ;LEFT SHIFT DATA PATTERN
4819  042526  077313                          SOB     R3,1$           ;BRANCH BACK IF R3 NOT 0
4820  042530  012701  000067                  MOV     #67,R1          ;PUT DATA PATTERN IN R1
4821  042534  012737  042542  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
4822  042542  000240                  12$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4823  042544  010110                          MOV     R1,(R0)         ;TRY TO SET ALL BITS IN MMR3
4824  042546  011002                          MOV     (R0),R2         ;READ MMR3 INTO R2
4825  042550  020102                          CMP     R1,R2           ;SEE IF ALL BITS GOT SET
4826  042552  001401                          BEQ     3$              ;BRANCH IF ALL BITS WERE SET
4827  042554  104016                          ERROR   16              ;MMR3 HELD WRONG DATA
4828
4829  042556  012737  042564  001110  3$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4830  042564  000240                  13$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
4831  042566  000005                          RESET                   ;ISSUE  'INIT'
4832  042570  011001                          MOV     (R0),R1         ;READ MMR3 INTO R2
4833  042572  001401                          BEQ     4$              ;BRANCH IF MMR3 INTO R2
4834  042574  104015                          ERROR   15              ;CAN'T CLEAR MMR3
4835  042576  012737  042450  001110  4$:     MOV     #20$,$LPERR     ;RESTORE LOOP POINTER TO FIRST INST.
4836
4837                                  .SBTTL  ********** ENTRY POINT 3 --- STARTING ADDRESS 210 **********
4838                                  .SBTTL  ********** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS **********
4839                                  ;*
4840                                  ;*      THIS GROUP OF TESTS IS USED TO CHECK ALL THE PAR'S AND
4841                                  ;*      PDR'S; THEIR ADDRESS DECODING ON DIRECT REFERENCES, THEIR
4842                                  ;*      READ/WRITE BITS, AND DUAL ADDRESSING WITHIN A GROUP OR BETWEEN
4843                                  ;*      TWO GROUPS.
4844                                  ;*
4845
4846
4847                                  .SBTTL          TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
4848                                  ;*      THESE NEXT SIX (6) TESTS VERIFY THAT THERE ARE 6 GROUPS OF
4849                                  ;*      16 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND
4850                                  ;*      WITHOUT TIMING OUT.  AT THIS POINT THE TEST IS NOT CONCERNED
4851                                  ;*      WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.
4852                                  ;*
4853                                  ;;*****************************************************************
4854                                  ;*TEST 15       READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
4855                                  ;*
4856                                  ;*      THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS
4857                                  ;*      'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND
4858                                  ;*      IF ANY OF THE 16 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O
4859                                  ;*      PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END OF THE TEST A
4860                                  ;*      SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.
4861                                  ;*
4862                                  ;*      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE ''TIMEOUT''
4863                                  ;*
4864                                  ;;*****************************************************************
4865  042604                          TST15:
```

```
                                        SCOPE
042604  000004
042606  012737  042740  001314        MOV     #TST16,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
042614                                ENTPT3:
042614  012737  042644  001106        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
042622  012737  042644  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
042630  012737  000015  001102        MOV     #15,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
042636  013737  001102  177570        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
042644  004737  024174              20$:  JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
042650  012737  024232  000004        MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
042656  012737  042670  001110        MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
042664  012700  172340                MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
042670  000240                    1$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
042672  011001                        MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
042674  062700  000002                ADD     #2,R0           ;POINT TO NEXT REGISTER
042700  022700  172376                CMP     #KDPAR7,R0      ;SEE IF KDPAR7 HAS BEEN TRIED
042704  103371                        BHIS    1$              ;BRANCH IF NOT DONE
042706  012737  042644  001110        MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
042714  012737  024516  000004        MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
042722  005737  001300                TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
042726  001404                        BEQ     TST16           ;;BRANCH TO NEXT TEST IF NO ERRORS
042730  013737  001300  001202        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
042736  104017                        ERROR   17              ;SUMMARY OF PAR ERRORS

                                    ;;**************************************************************
                                    ;*TEST 16       READ ALL SUPERVISOR PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
                                    ;;
                                    ;;      THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE ADDRESS
                                    ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
                                    ;;      TEST, AND IF ANY OF THE 16 SUPERVISOR ADDRESS REGISTERS TIME OUT
                                    ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
                                    ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
                                    ;;      'CPUER'.
                                    ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT''
                                    ;;
                                    ;;**************************************************************
042740                              TST16:
042740  000004                      SCOPE
042742  012737  043044  001314      MOV     #TST17,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                            ;TEST FOR ESCAPE ON PARITY ERRORS
042750  004737  024174            20$:  JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
042754  012737  024232  000004      MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
042762  012737  042774  001110      MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
042770  012700  172240              MOV     #SIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
042774  000240                    1$:   NOP                     ;THIS IS A SYNC POINT FOR SCOPING
042776  011001                      MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
043000  062700  000002              ADD     #2,R0           ;POINT TO NEXT REGISTER
043004  022700  172276              CMP     #SDPAR7,R0      ;SEE IF SDPAR7 HAS BEEN TRIED
043010  103371                      BHIS    1$              ;BRANCH IF NOT DONE
043012  012737  042750  001110      MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
043020  012737  024516  000004      MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
043026  005737  001300              TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
043032  001404                      BEQ     TST17           ;;BRANCH TO NEXT TEST IF NO ERRORS
043034  013737  001300  001202      MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
043042  104017                      ERROR   17              ;SUMMARY OF PAR ERRORS
```

```
                                ;;*****************************************************************
                                ;*TEST 17       READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
                                ;;
                                ;;      THIS TEST DOES A READ FROM ALL THE USER PAGE ADDRESS
                                ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
                                ;;      TEST, AND IF ANY OF THE 16 USER ADDRESS REGISTERS TIME OUT
                                ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
                                ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
                                ;;      'CPUER'.
                                ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT''
                                ;;
                                ;;*****************************************************************
043044                          TST17:
043044  000004                          SCOPE
043046  012737  043150  001314         MOV     #TST20,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
043054  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
043060  012737  024232  000004         MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
043066  012737  043100  001110         MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
043074  012700  177640                 MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
043100  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
043102  011001                         MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
043104  062700  000002                 ADD     #2,R0           ;POINT TO NEXT REGISTER
043110  022700  177676                 CMP     #UDPAR7,R0      ;SEE IF UDPAR7 HAS BEEN TRIED
043114  103371                         BHIS    1$              ;BRANCH IF NOT DONE
043116  012737  043054  001110         MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
043124  012737  024516  000004         MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
043132  005737  001300                 TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
043136  001404                         BEQ     TST20           ;;BRANCH TO NEXT TEST IF NO ERRORS
043140  013737  001300  001202         MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
043146  104017                         ERROR   17              ;SUMMARY OF PAR ERRORS

                                ;;*****************************************************************
                                ;*TEST 20       READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
                                ;;
                                ;;      THIS TEST DOES A READ FROM ALL THE KERNEL PAGE DESCRIPTOR
                                ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
                                ;;      TEST, AND IF ANY OF THE 16 KERNEL DESCRIPTOR REGISTERS TIME OUT
                                ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
                                ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
                                ;;      'CPUER'.
                                ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT''
                                ;;
                                ;;*****************************************************************
043150                          TST20:
043150  000004                          SCOPE
043152  012737  043254  001314         MOV     #TST21,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
043160  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
043164  012737  024232  000004         MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
043172  012737  043204  001110         MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
043200  012700  172300                 MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
043204  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
043206  011001                         MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
043210  062700  000002                 ADD     #2,R0           ;POINT TO NEXT REGISTER
043214  022700  172336                 CMP     #KDPDR7,R0      ;SEE IF KDPDR7 HAS BEEN TRIED
```

```
                        BHIS    1$              ;BRANCH IF NOT DONE
                        MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
                        MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
                        TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
                        BEQ     TST21           ;;BRANCH TO NEXT TEST IF NO ERRORS
                        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
                        ERROR   17              ;SUMMARY OF PDR ERRORS

                ;;**************************************************************
                ;*TEST 21        READ ALL SUPERVISOR PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
                ;;
                ;;      THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE DESCRIPTOR
                ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
                ;;      TEST, AND IF ANY OF THE 16 SUPERVISOR DESCRIPTOR REGISTERS TIME OUT
                ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
                ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
                ;;      'CPUER'.
                ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT''
                ;;
                ;;**************************************************************
                TST21:
                        SCOPE
                        MOV     #TST22,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
                        MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
                        MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
                        MOV     #SIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
                1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
                        MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
                        ADD     #2,R0           ;POINT TO NEXT REGISTER
                        CMP     #SDPDR7,R0      ;SEE IF SDPDR7 HAS BEEN TRIED
                        BHIS    1$              ;BRANCH IF NOT DONE
                        MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
                        MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
                        TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
                        BEQ     TST22           ;;BRANCH TO NEXT TEST IF NO ERRORS
                        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
                        ERROR   17              ;SUMMARY OF PDR ERRORS

                ;;**************************************************************
                ;*TEST 22        READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
                ;;
                ;;      THIS TEST DOES A READ FROM ALL THE USER PAGE DESCRIPTOR
                ;;      REGISTERS.  'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
                ;;      TEST, AND IF ANY OF THE 16 USER DESCRIPTOR REGISTERS TIME OUT
                ;;      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED.  AT THE END
                ;;      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
                ;;      'CPUER'.
                ;;      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT''
                ;;
                ;;**************************************************************
                TST22:
                        SCOPE
                        MOV     #TST23,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                ;TEST FOR ESCAPE ON PARITY ERRORS
```

```
5034  043370  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
5035  043374  012737  024232  000004          MOV     #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
5036  043402  012737  043414  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5037  043410  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
5038  043414  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5039  043416  011001                          MOV     (R0),R1         ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
5040  043420  062700  000002                  ADD     #2,R0           ;POINT TO NEXT REGISTER
5041  043424  022700  177636                  CMP     #UDPDR7,R0      ;SEE IF UDPDR7 HAS BEEN TRIED
5042  043430  103371                          BHIS    1$              ;BRANCH IF NOT DONE
5043  043432  012737  043370  001110          MOV     #20$,$LPERR     ;PUT LOOP ON ERROR POINTER TO START OF TEST
5044  043440  012737  024516  000004          MOV     #CPUER,ERRVEC   ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
5045  043446  005737  001300                  TST     ERRCNT          ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
5046  043452  001404                          BEQ     TST23           ;;BRANCH TO NEXT TEST IF NO ERRORS
5047  043454  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5048  043462  104017                          ERROR   17              ;SUMMARY OF PDR ERRORS
5049
5050                                  .SBTTL          TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
5051                                  ;*      THESE NEXT SIX (6) TESTS WILL CHECK FOR DUAL ADDRESSING WITHIN A
5052                                  ;*      GROUP OF PAR'S OR PDR'S.  FIRST ALL OF THE REGISTERS IN A GROUP
5053                                  ;*      ARE CLEARED AND READ TO SEE THAT THEY CAN EACH HOLD ZEROES, AND
5054                                  ;*      THAT THE DATA PATH DOES NOT HAVE A BIT STUCK AT ONE.
5055                                  ;*      THEN ONE REGISTER AT A TIME, WITHIN THAT GROUP, IS LOADED
5056                                  ;*      WITH A NEGATIVE ONE WHILE ALL REGISTERS ARE READ TO SEE
5057                                  ;*      THAT ONLY THE REGISTER UNDER TEST WAS NOT ZERO.
5058                                  ;*
5059                                  ;;*****************************************************************
5060                                  ;*TEST 23       DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
5061                                  ;*
5062                                  ;*      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE ADDRESS REGISTERS,
5063                                  ;*      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING
5064                                  ;*      WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME
5065                                  ;*      IS LOADED WITH A NEGATIVE ONE.  ALL KERNEL ADDRESS REGISTERS
5066                                  ;*      ARE NOW READ TO SEE THAT ONLY THE REGISTER UNDER TEST IS NON-
5067                                  ;*      ZERO.  INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM
5068                                  ;*      IS GIVEN AT THE END OF THIS TEST.
5069                                  ;*
5070                                  ;*      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5071                                  ;*      'DUALADR''.
5072                                  ;;*****************************************************************
5073  043464                          TST23:
5074  043464  000004                          SCOPE
5075  043466  012737  043674  001314          MOV     #TST24,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5076                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
5077  043474  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
5078  043500  012737  043526  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5079  043506  012705  172340                  MOV     #KIPAR0,R5      ;PUT ADDRESS OF FIRST PAR IN R5
5080  043512  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
5081  043516  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
5082  043522  012701  000020                  MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
5083  043526  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5084  043530  011002                          MOV     (R0),R2         ;READ PAR TO R2
5085  043532  001401                          BEQ     2$              ;BRANCH IF PAR IS 0
5086  043534  104020                          ERROR   20              ;PAR NOT ZERO
5087  043536  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
5088  043542  077107                          SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
5089                                  ;:
```

```
                                ;:  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
                                ;:  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
                                ;:  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
                                ;:
043544  012737  043556  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
043552  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IS R0
043556  012705  172340          3$:     MOV     #KIPAR0,R5      ;LOAD STARTING ADDRESS INTO R5
043562  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
043566  012701  172376                  MOV     #KDPAR7,R1      ;PUT KDPAR7 ADDRESS IN R1
043572  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
043574  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
043600  005037  001170          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
043604  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
043606  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
043610  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
                                                                ;AS THAT OF REGISTER UNDER TEST
043612  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
043614  004737  024352                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
043620  005237  001170          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
043624  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
043630  022701  172340                  CMP     #KIPAR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
043634  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
043636  062700  000002                  ADD     #2,R0           ;NOW LOAD NEXT REGISTER
043642  022700  172376                  CMP     #KDPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
043646  103343                          BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
043650  012737  043474  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
043656  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
043662  001404                          BEQ     TST24           ;;BRANCH TO NEXT TEST IF NO ERRORS
043664  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
043672  104021                          ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS

                                ;;******************************************************************
                                ;*TEST 24        DUAL ADDRESS SUPERVISOR PAGE ADDRESS REGISTERS, ON LOADING
                                ;:
                                ;:      THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE ADDRESS
                                ;:      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
                                ;:      I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
                                ;:      LOADED WITH A NEGATIVE ONE.  ALL SUPERVISOR ADDRESS REGISTERS
                                ;:      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
                                ;:      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
                                ;:      THE END OF THIS TEST.
                                ;:
                                ;:      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
                                ;:      'DUALADR'.
                                ;;******************************************************************
043674                          TST24:
043674  000004                          SCOPE
043676  012737  044104  001314          MOV     #TST25,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
043704  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
043710  012737  043736  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
043716  012705  172240                  MOV     #SIPAR0,R5      ;PUT ADDRESS OF FIRST PAR IN R5
043722  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
043726  012700  172240                  MOV     #SIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
043732  012701  000020                  MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
043736  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
```

```
5146  043740  011002                        MOV     (R0),R2         ;READ PAR TO R2
5147  043742  001401                        BEQ     2$              ;BRANCH IF PAR IS 0
5148  043744  104020                        ERROR   20              ;PAR NOT ZERO
5149  043746  062700  000002          2$:   ADD     #2,R0           ;POINT TO NEXT REGISTER
5150  043752  077107                        SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
5151                                  ;;
5152                                  ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5153                                  ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5154                                  ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
5155                                  ;;
5156  043754  012737  043766  001110        MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
5157  043762  012700  172240                MOV     #SIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IS R0
5158  043766  012705  172240          3$:   MOV     #SIPAR0,R5      ;LOAD STARTING ADDRESS INTO R5
5159  043772  004737  024156                JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
5160  043776  012701  172276                MOV     #SDPAR7,R1      ;PUT SDPAR7 ADDRESS IN R1
5161  044002  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5162  044004  012710  177777                MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
5163  044010  005037  001170          4$:   CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
5164  044014  011102                        MOV     (R1),R2         ;READ ALL REGISTERS
5165  044016  001406                        BEQ     6$              ;BRANCH IF REGISTER IS 0
5166  044020  020001                        CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
5167                                                                ;AS THAT OF REGISTER UNDER TEST
5168  044022  001402                        BEQ     5$              ;BRANCH IF ADDRESSES MATCH
5169  044024  004737  024352                JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
5170  044030  005237  001170          5$:   INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
5171  044034  162701  000002          6$:   SUB     #2,R1           ;POINT TO NEXT REGISTER
5172  044040  022701  172240                CMP     #SIPAR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
5173  044044  101761                        BLOS    4$              ;BRANCH IF MORE TO READ
5174  044046  062700  000002                ADD     #2,R0           ;NOW LOAD NEXT REGISTER
5175  044052  022700  172276                CMP     #SDPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
5176  044056  103343                        BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
5177  044060  012737  043704  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
5178  044066  005737  001300                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
5179  044072  001404                        BEQ     TST25           ;;BRANCH TO NEXT TEST IF NO ERRORS
5180  044074  013737  001300  001202        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
5181  044102  104021                        ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS
5182
5183                                  ;;**************************************************************
5184                                  ;*TEST 25       DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING
5185                                  ;;
5186                                  ;;      THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS
5187                                  ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
5188                                  ;;      I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
5189                                  ;;      LOADED WITH A NEGATIVE ONE.  ALL USER ADDRESS REGISTERS
5190                                  ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
5191                                  ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
5192                                  ;;      THE END OF THIS TEST.
5193                                  ;;
5194                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5195                                  ;;      'DUALADR'.
5196                                  ;;**************************************************************
5197  044104                          TST25:
5198  044104  000004                        SCOPE
5199  044106  012737  044314  001314        MOV     #TST26,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5200                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
5201  044114  004737  024174          20$:  JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
```

```
5202  044120  012737  044146  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5203  044126  012705  177640                  MOV     #UIPAR0,R5      ;PUT ADDRESS OF FIRST PAR IN R5
5204  044132  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
5205  044136  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IN R0
5206  044142  012701  000020                  MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
5207  044146  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5208  044150  011002                          MOV     (R0),R2         ;READ PAR TO R2
5209  044152  001401                          BEQ     2$              ;BRANCH IF PAR IS 0
5210  044154  104020                          ERROR   20              ;PAR NOT ZERO
5211  044156  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
5212  044162  077107                          SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
5213                                  ;;
5214                                  ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5215                                  ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5216                                  ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
5217                                  ;;
5218  044164  012737  044176  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
5219  044172  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST PAR IS R0
5220  044176  012705  177640          3$:     MOV     #UIPAR0,R5      ;LOAD STARTING ADDRESS INTO R5
5221  044202  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
5222  044206  012701  177676                  MOV     #UDPAR7,R1      ;PUT UDPAR7 ADDRESS IN R1
5223  044212  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5224  044214  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
5225  044220  005037  001170          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
5226  044224  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
5227  044226  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
5228  044230  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
5229                                                                  ;AS THAT OF REGISTER UNDER TEST
5230  044232  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
5231  044234  004737  024352                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
5232  044240  005237  001170          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
5233  044244  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
5234  044250  022701  177640                  CMP     #UIPAR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
5235  044254  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
5236  044256  062700  000002                  ADD     #2,R0           ;NOW LOAD NEXT REGISTER
5237  044262  022700  177676                  CMP     #UDPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
5238  044266  103343                          BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
5239  044270  012737  044114  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
5240  044276  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
5241  044302  001404                          BEQ     TST26           ;;BRANCH TO NEXT TEST IF NO ERRORS
5242  044304  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
5243  044312  104021                          ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS
5244
5245                                  ;;**********************************************************************
5246                                  ;*TEST 26        DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING
5247                                  ;;
5248                                  ;;      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR
5249                                  ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
5250                                  ;;      I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
5251                                  ;;      LOADED WITH A NEGATIVE ONE.  ALL KERNEL DESCRIPTOR REGISTERS
5252                                  ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
5253                                  ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
5254                                  ;;      THE END OF THIS TEST.
5255                                  ;;
5256                                  ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5257                                  ;;      'DUALADR'.
```

```
                                ;;*****************************************************************
044314                          TST26:
                                        SCOPE
044314  000004
044316  012737  044524  001314          MOV     #TST27,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
044324  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
044330  012737  044356  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
044336  012705  172300                  MOV     #KIPDR0,R5      ;PUT ADDRESS OF FIRST PDR IN R5
044342  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
044346  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
044352  012701  000020                  MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
044356  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
044360  011002                          MOV     (R0),R2         ;READ PDR TO R2
044362  001401                          BEQ     2$              ;BRANCH IF PDR IS 0
044364  104020                          ERROR   20              ;PDR NOT ZERO
044366  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
044372  077107                          SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
                                ;;
                                ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
                                ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
                                ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
                                ;;
044374  012737  044406  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
044402  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
044406  012705  172300          3$:     MOV     #KIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
044412  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
044416  012701  172336                  MOV     #KDPDR7,R1      ;PUT KDPDR7 ADDRESS IN R1
044422  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
044424  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
044430  005037  001170          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
044434  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
044436  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
044440  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
                                                                ;AS THAT OF REGISTER UNDER TEST
044442  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
044444  004737  024352                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
044450  005237  001170          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
044454  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
044460  022701  172300                  CMP     #KIPDR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
044464  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
044466  062700  000002                  ADD     #2,R0           ;NOW LOAD NEXT REGISTER
044472  022700  172336                  CMP     #KDPDR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
044476  103343                          BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
044500  012737  044324  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
044506  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
044512  001404                          BEQ     TST27           ;;BRANCH TO NEXT TEST IF NO ERRORS
044514  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PDR OUT
044522  104021                          ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS

                                ;;*****************************************************************
                                ;*TEST 27        DUAL ADDRESS SUPERVISOR PAGE DESCRIPTOR REGISTERS, ON LOADING
                                ;;
                                ;;      THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE DESCRIPTOR
                                ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
                                ;;      I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
                                ;;      LOADED WITH A NEGATIVE ONE.  ALL SUPERVISOR DESCRIPTOR REGISTERS
```

```
                                ;:      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
                                ;:      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
                                ;:      THE END OF THIS TEST.
                                ;:
                                ;:      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
                                ;:      'DUALADR'.
                                ;;**********************************************************************
044524                          TST27:
044524  000004                          SCOPE
044526  012737  044734  001314          MOV     #TST30,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
044534  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
044540  012737  044566  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
044546  012705  172200                  MOV     #SIPDR0,R5      ;PUT ADDRESS OF FIRST PDR IN R5
044552  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
044556  012700  172200                  MOV     #SIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
044562  012701  000020                  MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
044566  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
044570  011002                          MOV     (R0),R2         ;READ PDR TO R2
044572  001401                          BEQ     2$              ;BRANCH IF PDR IS 0
044574  104020                          ERROR   20              ;PDR NOT ZERO
044576  062700  000002          2$:     ADD     #2,R0           ;POINT TO NEXT REGISTER
044602  077107                          SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
                                ;:
                                ;:  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
                                ;:  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
                                ;:  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
                                ;:
044604  012737  044616  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
044612  012700  172200                  MOV     #SIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
044616  012705  172200          3$:     MOV     #SIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
044622  004737  024156                  JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
044626  012701  172236                  MOV     #SDPDR7,R1      ;PUT SDPDR7 ADDRESS IN R1
044632  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
044634  012710  177777                  MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
044640  005037  001170          4$:     CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
044644  011102                          MOV     (R1),R2         ;READ ALL REGISTERS
044646  001406                          BEQ     6$              ;BRANCH IF REGISTER IS 0
044650  020001                          CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
                                                                ;AS THAT OF REGISTER UNDER TEST
044652  001402                          BEQ     5$              ;BRANCH IF ADDRESSES MATCH
044654  004737  024352                  JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
044660  005237  001170          5$:     INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
044664  162701  000002          6$:     SUB     #2,R1           ;POINT TO NEXT REGISTER
044670  022701  172200                  CMP     #SIPDR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
044674  101761                          BLOS    4$              ;BRANCH IF MORE TO READ
044676  062700  000002                  ADD     #2,R0           ;NOW LOAD NEXT REGISTER
044702  022700  172236                  CMP     #SDPDR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
044706  103343                          BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
044710  012737  044534  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
044716  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
044722  001404                          BEQ     TST30           ;;BRANCH TO NEXT TEST IF NO ERRORS
044724  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PDR OUT
044732  104021                          ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS

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

```
                                        ;*TEST 30       DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING
                                        ;;
                                        ;;      THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR
                                        ;;      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO.  THEN, STARTING WITH
                                        ;;      I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
                                        ;;      LOADED WITH A NEGATIVE ONE.  ALL USER DESCRIPTOR REGISTERS
                                        ;;      ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
                                        ;;      INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
                                        ;;      THE END OF THIS TEST.
                                        ;;
                                        ;;      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
                                        ;;      'DUALADR'.
                                        ;;*****************************************************************
044734                                  TST30:
044734  000004                                  SCOPE
044736  012737  045144  001314                 MOV     #TST31,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
044744  004737  024174                  20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
044750  012737  044776  001110                 MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
044756  012705  177600                         MOV     #UIPDR0,R5      ;PUT ADDRESS OF FIRST PDR IN R5
044762  004737  024156                         JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
044766  012700  177600                         MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IN R0
044772  012701  000020                         MOV     #20,R1          ;BRANCH COUNT IS 16 DECIMAL
044776  000240                          1$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
045000  011002                                 MOV     (R0),R2         ;READ PDR TO R2
045002  001401                                 BEQ     2$              ;BRANCH IF PDR IS 0
045004  104020                                 ERROR   20              ;PDR NOT ZERO
045006  062700  000002                  2$:    ADD     #2,R0           ;POINT TO NEXT REGISTER
045012  077107                                 SOB     R1,1$           ;BRANCH BACK TO 1$ 15 TIMES
                                        ;;
                                        ;;  NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
                                        ;;  REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
                                        ;;  IS NON-ZERO. (DROPPED BITS  WILL BE FOUND IN NEXT TEST.)
                                        ;;
045014  012737  045026  001110                 MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
045022  012700  177600                         MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
045026  012705  177600                  3$:    MOV     #UIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
045032  004737  024156                         JSR     PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
045036  012701  177636                         MOV     #UDPDR7,R1      ;PUT UDPDR7 ADDRESS IN R1
045042  000240                                 NOP                     ;THIS IS A SYNC POINT FOR SCOPING
045044  012710  177777                         MOV     #-1,(R0)        ;LOAD REGISTER UNDER TEST
045050  005037  001170                  4$:    CLR     $TMP0           ;FLAG TO INDICATE THERE WAS A MATCH
045054  011102                                 MOV     (R1),R2         ;READ ALL REGISTERS
045056  001406                                 BEQ     6$              ;BRANCH IF REGISTER IS 0
045060  020001                                 CMP     R0,R1           ;IS ADDRESS OF NON-ZERO REGISTER SAME
                                                                        ;AS THAT OF REGISTER UNDER TEST
045062  001402                                 BEQ     5$              ;BRANCH IF ADDRESSES MATCH
045064  004737  024352                         JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
045070  005237  001170                  5$:    INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
045074  162701  000002                  6$:    SUB     #2,R1           ;POINT TO NEXT REGISTER
045100  022701  177600                         CMP     #UIPDR0,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
045104  101761                                 BLOS    4$              ;BRANCH IF MORE TO READ
045106  062700  000002                         ADD     #2,R0           ;NOW LOAD NEXT REGISTER
045112  022700  177636                         CMP     #UDPDR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
045116  103343                                 BHIS    3$              ;BRANCH IF MORE REGISTERS TO TEST
045120  012737  044744  001110                 MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
```

```
                                        TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
                                        BEQ     TST31           ;;BRANCH TO NEXT TEST IF NO ERRORS
                                        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PDR OUT
                                        ERROR   21              ;SUMMARY OF DUAL ADDRESSING ERRORS

                                .SBTTL          TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
                                ;*      THESE NEXT SIX (6) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
                                ;*      THAT MAKE UP THE PAR'S AND PDR'S.  THE REGISTERS ARE LOADED WITH
                                ;*      ZERO AND MODIFIED BY "401" UNTIL 177777 IS REACHED IN EACH ONE
                                ;*      (THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME).  THE BITS THAT
                                ;*      ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
                                ;*      OF THE DATA COMPARE.  A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
                                ;*      GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.
                                ;*
                                ;;*********************************************************************
                                ;*TEST 31       COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
                                ;*
                                ;*      THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
                                ;*      REGISTERS.  SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
                                ;*      TO BE MASKED OUT OF THE COMPARE.  IF THE COUNT PATTERN DOES
                                ;*      NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
                                ;*      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
                                ;*      SUMMARY OF THE ERRORS IS GIVEN.
                                ;*
                                ;*      ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
                                ;*      SUBROUTINE 'PARCOUNT".
                                ;*
                                ;;*********************************************************************
045144                          TST31:
045144  000004                          SCOPE
045146  012737  045274  001314          MOV     #TST32,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
045154  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
045162  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
045166  012737  045202  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
045174  005001                          CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
045176  012700  172340          1$:     MOV     #KIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
045202  000240                  2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
045204  010110                          MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
045206  011002                          MOV     (R0),R2         ;READ REGISTER BACK TO R2
045210  010104                          MOV     R1,R4           ;PUT PATTERN IS R4
045212  020402                          CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
045214  001402                          BEQ     3$              ;BRANCH IF DATA IS GOOD
045216  004737  024426                  JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
045222  062700  000002          3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
045226  022700  172376                  CMP     #KDPAR7,R0      ;SEE IF YOU PASSED THE KDPAR7 PAR
045232  103363                          BHIS    2$              ;BRANCH IF MORE PAR''S TO TEST
045234  022701  177777                  CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
045240  001403                          BEQ     4$              ;BRANCH IF COUNT IS 177777
045242  062701  000401                  ADD     #401,R1         ;INCREASE COUNT PATTERN
045246  000753                          BR      1$              ;BRANCH TO CONTINUE TEST
045250  012737  045162  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
045256  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
045262  001404                          BEQ     TST32           ;;BRANCH TO NEXT TEST IF NO ERRORS
045264  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
045272  104022                          ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES
```

```
                        ;;*****************************************************************
                        ;*TEST 32        COUNT PATTERN IN SUPERVISOR PAGE ADDRESS REGISTERS
                        ;;
                        ;;      THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE ADDRESS
                        ;;      REGISTERS.  SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
                        ;;      TO BE MASKED OUT OF THE COMPARE.  IF THE COUNT PATTERN DOES
                        ;;      NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
                        ;;      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
                        ;;      SUMMARY OF THE ERRORS IS GIVEN.
                        ;;
                        ;;      ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
                        ;;      SUBROUTINE 'PARCOUNT''.
                        ;;*****************************************************************
045274                  TST32:
045274  000004                  SCOPE
045276  012737  045424  001314        MOV     #TST33,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
045304  012737  000012  001204        MOV     #12,$TIMES      ;;DO 12 ITERATIONS
045312  004737  024174        20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
045316  012737  045332  001110        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
045324  005001                        CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
045326  012700  172240        1$:     MOV     #SIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
045332  000240        2$:     NOP                             ;THIS IS A SYNC POINT FOR SCOPING
045334  010110                        MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
045336  011002                        MOV     (R0),R2         ;READ REGISTER BACK TO R2
045340  010104                        MOV     R1,R4           ;PUT PATTERN IS R4
045342  020402                        CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
045344  001402                        BEQ     3$              ;BRANCH IF DATA IS GOUD
045346  004737  024426                JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
045352  062700  000002        3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
045356  022700  172276                CMP     #SDPAR7,R0      ;SEE IF YOU PASSED THE SDPAR7 PAR
045362  103363                        BHIS    2$              ;BRANCH IF MORE PAR''S TO TEST
045364  022701  177777                CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
045370  001403                        BEQ     4$              ;BRANCH IF COUNT IS 177777
045372  062701  000401                ADD     #401,R1         ;INCREASE COUNT PATTERN
045376  000753                        BR      1$              ;BRANCH TO CONTINUE TEST
045400  012737  045312  001110  4$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
045406  005737  001300                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
045412  001404                        BEQ     TST33           ;;BRANCH TO NEXT TEST IF NO ERRORS
045414  013737  001300  001202        MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
045422  104022                        ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES

                        ;;*****************************************************************
                        ;*TEST 33        COUNT PATTERN IN USER PAGE ADDRESS REGISTERS
                        ;;
                        ;;      THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE ADDRESS
                        ;;      REGISTERS.  SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
                        ;;      TO BE MASKED OUT OF THE COMPARE.  IF THE COUNT PATTERN DOES
                        ;;      NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
                        ;;      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
                        ;;      SUMMARY OF THE ERRORS IS GIVEN.
                        ;;
                        ;;      ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
                        ;;      SUBROUTINE 'PARCOUNT''.
```

```
5538                                        ;;
5539                                        ;;*****************************************************************
5540  045424                                TST33:
5541  045424  000004                              SCOPE
5542  045426  012737  045554  001314              MOV     #TST34,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5543                                                                      ;TEST FOR ESCAPE ON PARITY ERRORS
5544  045434  012737  000012  001204              MOV     #12,$TIMES      ;;DO 12 ITERATIONS
5545  045442  004737  024174              20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
5546  045446  012737  045462  001110              MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5547  045454  005001                              CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
5548  045456  012700  177640              1$:     MOV     #UIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
5549  045462  000240                      2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5550  045464  010110                              MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
5551  045466  011002                              MOV     (R0),R2         ;READ REGISTER BACK TO R2
5552  045470  010104                              MOV     R1,R4           ;PUT PATTERN IS R4
5553  045472  020402                              CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
5554  045474  001402                              BEQ     3$              ;BRANCH IF DATA IS GOOD
5555  045476  004737  024426                      JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
5556  045502  062700  000002              3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
5557  045506  022700  177676                      CMP     #UDPAR7,R0      ;SEE IF YOU PASSED THE UDPAR7 PAR
5558  045512  103363                              BHIS    2$              ;BRANCH IF MORE PAR''S TO TEST
5559  045514  022701  177777                      CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
5560  045520  001403                              BEQ     4$              ;BRANCH IF COUNT IS 177777
5561  045522  062701  000401                      ADD     #401,R1         ;INCREASE COUNT PATTERN
5562  045526  000753                              BR      1$              ;BRANCH TO CONTINUE TEST
5563  045530  012737  045442  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5564  045536  005737  001300                      TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
5565  045542  001404                              BEQ     TST34           ;;BRANCH TO NEXT TEST IF NO ERRORS
5566  045544  013737  001300  001202              MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5567  045552  104022                              ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES
5568
5569                                        ;;*****************************************************************
5570                                        ;*TEST 34        COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS
5571                                        ;*
5572                                        ;*      THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE DESCRIPTOR
5573                                        ;*      REGISTERS.  SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED
5574                                        ;*      AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
5575                                        ;*      OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT TO THE
5576                                        ;*      DESCRIPTOR REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
5577                                        ;*      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
5578                                        ;*      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
5579                                        ;*      SUMMARY OF THE ERRORS IS GIVEN.
5580                                        ;*
5581                                        ;;*****************************************************************
5582  045554                                TST34:
5583  045554  000004                              SCOPE
5584  045556  012737  045710  001314              MOV     #TST35,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5585                                                                      ;TEST FOR ESCAPE ON PARITY ERRORS
5586  045564  012737  000012  001204              MOV     #12,$TIMES      ;;DO 12 ITERATIONS
5587  045572  004737  024174              20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
5588  045576  012737  045612  001110              MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5589  045604  005001                              CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
5590  045606  012700  172300              1$:     MOV     #KIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
5591  045612  000240                      2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5592  045614  010110                              MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
5593  045616  011002                              MOV     (R0),R2         ;READ REGISTER BACK TO R2
```

```
5594  045620  010104                          MOV     R1,R4           ;PUT PATTERN IS R4
5595  045622  042704  100360                  BIC     #100360,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
5596  045626  020402                          CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
5597  045630  001402                          BEQ     3$              ;BRANCH IF DATA IS GOOD
5598  045632  004737  024426                  JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
5599  045636  062700  000002          3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
5600  045642  022700  172336                  CMP     #KDPDR7,R0      ;SEE IF YOU PASSED THE KDPDR7 PDR
5601  045646  103361                          BHIS    2$              ;BRANCH IF MORE PDR''S TO TEST
5602  045650  022701  177777                  CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
5603  045654  001403                          BEQ     4$              ;BRANCH IF COUNT IS 177777
5604  045656  062701  000401                  ADD     #401,R1         ;INCREASE COUNT PATTERN
5605  045662  000751                          BR      1$              ;BRANCH TO CONTINUE TEST
5606  045664  012737  045572  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5607  045672  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
5608  045676  001404                          BEQ     TST35           ;;BRANCH TO NEXT TEST IF NO ERRORS
5609  045700  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5610  045706  104022                          ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES
5611
5612                                  ;;**********************************************************************
5613                                  ;*TEST 35        COUNT PATTERN IN SUPERVISOR PAGE DESCRIPTOR REGISTERS
5614                                  ;;
5615                                  ;;      THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE DESCRIPTOR
5616                                  ;;      REGISTERS.  SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED
5617                                  ;;      AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
5618                                  ;;      OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT TO THE
5619                                  ;;      DESCRIPTOR REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
5620                                  ;;      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
5621                                  ;;      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
5622                                  ;;      SUMMARY OF THE ERRORS IS GIVEN.
5623                                  ;;
5624                                  ;;**********************************************************************
5625  045710                          TST35:
5626  045710  000004                          SCOPE
5627  045712  012737  046044  001314          MOV     #TST36,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5628                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
5629  045720  012737  000012  001204          MOV     #12,$TIMES      ;;DO 12 ITERATIONS
5630  045726  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
5631  045732  012737  045746  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5632  045740  005001                          CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
5633  045742  012700  172200          1$:     MOV     #SIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
5634  045746  000240                  2$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5635  045750  010110                          MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
5636  045752  011002                          MOV     (R0),R2         ;READ REGISTER BACK TO R2
5637  045754  010104                          MOV     R1,R4           ;PUT PATTERN IS R4
5638  045756  042704  100360                  BIC     #100360,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
5639  045762  020402                          CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
5640  045764  001402                          BEQ     3$              ;BRANCH IF DATA IS GOOD
5641  045766  004737  024426                  JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
5642  045772  062700  000002          3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
5643  045776  022700  172236                  CMP     #SDPDR7,R0      ;SEE IF YOU PASSED THE SDPDR7 PDR
5644  046002  103361                          BHIS    2$              ;BRANCH IF MORE PDR''S TO TEST
5645  046004  022701  177777                  CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
5646  046010  001403                          BEQ     4$              ;BRANCH IF COUNT IS 177777
5647  046012  062701  000401                  ADD     #401,R1         ;INCREASE COUNT PATTERN
5648  046016  000751                          BR      1$              ;BRANCH TO CONTINUE TEST
5649  046020  012737  045726  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                                                TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
                                                BEQ     TST36           ;;BRANCH TO NEXT TEST IF NO ERRORS
                                                MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
                                                ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES

                                        ;;******************************************************************
                                        ;*TEST 36       COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS
                                        ;:
                                        ;:      THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR
                                        ;:      REGISTERS.  SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED
                                        ;:      AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
                                        ;:      OUT OF THE DATA COMPARE.  THESE BITS ARE STILL SENT TO THE
                                        ;:      DESCRIPTOR REGISTER TO FIND BAD ETCHES.  IF THE COUNT PATTERN
                                        ;:      DOES NOT MATCH THE DATA RECEVIED, THE REGISTER ADDRESS, DATA
                                        ;:      PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
                                        ;:      SUMMARY OF THE ERRORS IS GIVEN.
                                        ;:
                                        ;;******************************************************************
046044                                  TST36:
                                                SCOPE
046044  000004
046046  012737  046200  001314                  MOV     #TST37,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
046054  012737  000012  001204                  MOV     #12,$TIMES      ;;DO 12 ITERATIONS
046062  004737  024174          20$:    JSR     PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
046066  012737  046102  001110                  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
046074  005001                                  CLR     R1              ;CLEAR REGISTER TO HOLD COUNT PATTERN
046076  012700  177600          1$:     MOV     #UIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER IS R0
046102  000240          2$:     NOP                             ;THIS IS A SYNC POINT FOR SCOPING
046104  010110                                  MOV     R1,(R0)         ;LOAD COUNT INTO REGISTER
046106  011002                                  MOV     (R0),R2         ;READ REGISTER BACK TO R2
046110  010104                                  MOV     R1,R4           ;PUT PATTERN IS R4
046112  042704  100360                          BIC     #100360,R4      ;CLEAR BITS NOT FOUND IN REGISTER.
046116  020402                                  CMP     R4,R2           ;SEE IF DATA MATCHES PATTERN
046120  001402                                  BEQ     3$              ;BRANCH IF DATA IS GOOD
046122  004737  024426                          JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
046126  062700  000002          3$:     ADD     #2,R0           ;POINT TO  NEXT REGISTER
046132  022700  177636                  CMP     #UDPDR7,R0      ;SEE IF YOU PASSED THE UDPDR7 PDR
046136  103361                          BHIS    2$              ;BRANCH IF MORE PDR''S TO TEST
046140  022701  177777                  CMP     #177777,R1      ;SEE IF COUNT HAS REACHED 177777
046144  001403                          BEQ     4$              ;BRANCH IF COUNT IS 177777
046146  062701  000401                  ADD     #401,R1         ;INCREASE COUNT PATTERN
046152  000751                          BR      1$              ;BRANCH TO CONTINUE TEST
046154  012737  046062  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
046162  005737  001300                  TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
046166  001404                          BEQ     TST37           ;;BRANCH TO NEXT TEST IF NO ERRORS
046170  013737  001300  001202          MOV     ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
046176  104022                          ERROR   22              ;SUMMARY OF COUNT PATTERN FAILURES

                                        .SBTTL          TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
                                        ;*      THESE NEXT SIX (6) TESTS ARE USED TO TEST THE LOGIC THAT
                                        ;*      ALLOWS BYTE ADDRESSING OF THE PAR'S AND PDR'S.  IN EACH
                                        ;*      CASE THE I-SPACE REGISTER 0 IS USED, SINCE IT IS REALLY
                                        ;*      THE WRITE PULSE TO THE REGISTER SET THAT IS BEING TESTED.
                                        ;*      IT IS ASSUMED THAT IF ONE REGISTER OF A GROUP FUNCTIONS
                                        ;*      PROPERLY THAT THEY ALL WILL, SINCE ALL REGISTERS HAVE BEEN
                                        ;*      BIT TESTED EARLIER.
```

```
                                ;*
                                ;;*****************************************************************
                                ;*TEST 37       BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS
                                ;*
                                ;*      THIS TEST CHECKS THE 'SAPA WRITE LOBYTE' AND 'SAPA WRITE HIBYTE'
                                ;*      SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
                                ;*      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
                                ;*      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
                                ;*
                                ;;*****************************************************************
046200                          TST37:
046200  000004                          SCOPE
046202  012737  046316  001314          MOV     #TST40,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
046210  012737  046226  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
046216  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
046222  012700  172340                  MOV     #KIPAR0,R0      ;PUT ADDRESS OF REGISTER IN R0
046226  005037  172340          11$:    CLR     KIPAR0          ;CLEAR THE REGISTER UNDER TEST
046232  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
046234  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
046240  013701  172340                  MOV     KIPAR0,R1       ;READ REGISTER INTO R1
046244  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
046246  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
046250  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
046252  012737  046270  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
046260  012700  172341                  MOV     #KIPAR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
046264  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
046270  000240          12$:    NOP                             ;THIS IS A SYNC POINT FOR SCOPING
046272  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
046276  013701  172340                  MOV     KIPAR0,R1       ;READ REGISTER INTO R1
046302  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
046304  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
046306  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
046310  012737  046210  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                ;;*****************************************************************
                                ;*TEST 40       BYTE ADDRESSING OF SUPERVISOR PAGE ADDRESS REGISTERS
                                ;;
                                ;;      THIS TEST CHECKS THE 'SAPB WRITE LOBYTE' AND 'SAPB WRITE HIBYTE'
                                ;;      SIGNAL GENERATION FOR SUPERVISOR PAGE ADDRESS REGISTERS.
                                ;;      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
                                ;;      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
                                ;;
                                ;;*****************************************************************
046316                          TST40:
046316  000004                          SCOPE
046320  012737  046434  001314          MOV     #TST41,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
046326  012737  046344  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
046334  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
046340  012700  172240                  MOV     #SIPAR0,R0      ;PUT ADDRESS OF REGISTER IN R0
046344  005037  172240          11$:    CLR     SIPAR0          ;CLEAR THE REGISTER UNDER TEST
046350  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
046352  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
046356  013701  172240                  MOV     SIPAR0,R1       ;READ REGISTER INTO R1
```

```
5762  046362  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
5763  046364  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
5764  046366  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5765  046370  012737  046406  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
5766  046376  012700  172241                  MOV     #SIPAR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
5767  046402  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
5768  046406  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5769  046410  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
5770  046414  013701  172240                  MOV     SIPAR0,R1       ;READ REGISTER INTO R1
5771  046420  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
5772  046422  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
5773  046424  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5774  046426  012737  046326  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5775
5776
5777                                  ;;**********************************************************************
5778                                  ;*TEST 41        BYTE ADDRESSING OF USER PAGE ADDRESS REGISTERS
5779                                  ;:
5780                                  ;:      THIS TEST CHECKS THE 'SAPC WRITE LOBYTE' AND 'SAPC WRITE HIBYTE'
5781                                  ;:      SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTEPS.
5782                                  ;:      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
5783                                  ;:      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
5784                                  ;:
5785                                  ;;**********************************************************************
5786  046434                          TST41:
5787  046434  000004                          SCOPE
5788  046436  012737  046552  001314          MOV     #TST42,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5789                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
5790  046444  012737  046462  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
5791  046452  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
5792  046456  012700  177640                  MOV     #UIPAR0,R0      ;PUT ADDRESS OF REGISTER IN R0
5793  046462  005037  177640          11$:    CLR     UIPAR0          ;CLEAR THE REGISTER UNDER TEST
5794  046466  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5795  046470  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
5796  046474  013701  177640                  MOV     UIPAR0,R1       ;READ REGISTER INTO R1
5797  046500  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
5798  046502  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
5799  046504  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5800  046506  012737  046524  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
5801  046514  012700  177641                  MOV     #UIPAR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
5802  046520  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
5803  046524  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5804  046526  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
5805  046532  013701  177640                  MOV     UIPAR0,R1       ;READ REGISTER INTO R1
5806  046536  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
5807  046540  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
5808  046542  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5809  046544  012737  046444  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5810
5811
5812                                  ;;**********************************************************************
5813                                  ;*TEST 42        BYTE ADDRESSING OF KERNEL PAGE DESCRIPTOR REGISTERS
5814                                  ;:
5815                                  ;:      THIS TEST CHECKS THE 'SAPD WRITE LOBYTE' AND 'SAPD WRITE HIBYTE'
5816                                  ;:      SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
5817                                  ;:      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
```

```
                                        ;;      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
                                        ;;
                                        ;;*****************************************************************
046552                                  TST42:
046552  000004                                  SCOPE
046554  012737  046670  001314                  MOV     #TST43,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
046562  012737  046600  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
046570  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
046574  012700  172300                  MOV     #KIPDR0,R0      ;PUT ADDRESS OF REGISTER IN R0
046600  005037  172300          11$:    CLR     KIPDR0          ;CLEAR THE REGISTER UNDER TEST
046604  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
046606  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
046612  013701  172300                  MOV     KIPDR0,R1       ;READ REGISTER INTO R1
046616  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
046620  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
046622  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
046624  012737  046642  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
046632  012700  172301                  MOV     #KIPDR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
046636  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
046642  000240          12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
046644  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
046650  013701  172300                  MOV     KIPDR0,R1       ;READ REGISTER INTO R1
046654  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
046656  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
046660  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
046662  012737  046562  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

                                        ;;*****************************************************************
                                        ;*TEST 43        BYTE ADDRESSING OF SUPERVISOR PAGE DESCRIPTOR REGISTERS
                                        ;;
                                        ;;      THIS TEST CHECKS THE 'SAPE WRITE LOBYTE' AND 'SAPE WRITE HIBYTE'
                                        ;;      SIGNAL GENERATION FOR SUPERVISOR PAGE ADDRESS REGISTERS.
                                        ;;      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
                                        ;;      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
                                        ;;
                                        ;;*****************************************************************
046670                                  TST43:
046670  000004                                  SCOPE
046672  012737  047006  001314                  MOV     #TST44,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
046700  012737  046716  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
046706  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
046712  012700  172200                  MOV     #SIPDR0,R0      ;PUT ADDRESS OF REGISTER IN R0
046716  005037  172200          11$:    CLR     SIPDR0          ;CLEAR THE REGISTER UNDER TEST
046722  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
046724  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
046730  013701  172200                  MOV     SIPDR0,R1       ;READ REGISTER INTO R1
046734  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
046736  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
046740  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
046742  012737  046760  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
046750  012700  172201                  MOV     #SIPDR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
046754  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
046760  000240          12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
```

```
5874  046762  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
5875  046766  013701  172200                  MOV     SIPDR0,R1       ;READ REGISTER INTO R1
5876  046772  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
5877  046774  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
5878  046776  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5879  047000  012737  046700  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5880
5881
5882                                  ;;*****************************************************************
5883                                  ;*TEST 44       BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS
5884                                  ;;
5885                                  ;;      THIS TEST CHECKS THE 'SAPF WRITE LOBYTE' AND 'SAPF WRITE HIBYTE'
5886                                  ;;      SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
5887                                  ;;      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
5888                                  ;;      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
5889                                  ;;
5890                                  ;;*****************************************************************
5891  047006                          TST44:
5892  047006  000004                          SCOPE
5893  047010  012737  047124  001314          MOV     #TST45,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5894                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
5895  047016  012737  047034  001110  20$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
5896  047024  012702  000015                  MOV     #15,R2          ;PUT EXPECTED DATA IN R2
5897  047030  012700  177600                  MOV     #UIPDR0,R0      ;PUT ADDRESS OF REGISTER IN R0
5898  047034  005037  177600          11$:    CLR     UIPDR0          ;CLEAR THE REGISTER UNDER TEST
5899  047040  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5900  047042  112710  172015                  MOVB    #172015,(R0)    ;LOAD LOWER BYTE OF REGISTER
5901  047046  013701  177600                  MOV     UIPDR0,R1       ;READ REGISTER INTO R1
5902  047052  020102                          CMP     R1,R2           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
5903  047054  001401                          BEQ     1$              ;BRANCH IF DATA MATCHES
5904  047056  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5905  047060  012737  047076  001110  1$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
5906  047066  012700  177601                  MOV     #UIPDR0+1,R0    ;POINT TO UPPER BYTE OF REGISTER
5907  047072  012702  052015                  MOV     #052015,R2      ;LOAD EXPECTED DATA IN R2
5908  047076  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
5909  047100  112710  000124                  MOVB    #124,(R0)       ;WRITE THE UPPER BYTE OF REGISTER
5910  047104  013701  177600                  MOV     UIPDR0,R1       ;READ REGISTER INTO R1
5911  047110  020102                          CMP     R1,R2           ;SEE IF REGISTER HOLDS CORRECT DATA
5912  047112  001401                          BEQ     2$              ;BRANCH TO EXIT IF DATA RIGHT
5913  047114  104023                          ERROR   23              ;DIDN'T LOAD CORRECT BYTE
5914  047116  012737  047016  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
5915
5916
5917                                  .SBTTL          DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS
5918                                  ;;*****************************************************************
5919                                  ;*TEST 45       DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
5920                                  ;*
5921                                  ;*      THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING BETWEEN
5922                                  ;*      GROUPS OF PAR'S AND PDR'S.  THAT IS, WHEN YOU REFERENCE A KERNEL
5923                                  ;*      I-SPACE PAR YOU ARE REALLY REFERENCING THAT PAR.  FIRST EACH
5924                                  ;*      I-SPACE PAR0 OR PDR0 IS LOADED WITH A UNIQUE NUMBER 0-12 AND
5925                                  ;*      THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
5926                                  ;*      EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
5927                                  ;*      ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
5928                                  ;*      BE TESTED HERE.
5929                                  ;*
```

```
5930
5931  047124                                  ;;*****************************************************************
5932  047124  000004                          TST45:
5933  047126  012737  047226  001314                  SCOPE
                                                MOV     #TST46,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5934                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
5935  047134  012737  047150  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
5936  047142  005000                          CLR     R0              ;THIS WILL HOLD THE INDEX OF EACH REGISTER
5937                                                                    ;AND THE COUNT LOADED
5938  047144  012704  000006                  MOV     #6,R4           ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
5939  047150  010070  001320          1$:     MOV     R0,@PARTAB(R0)  ;LOAD PAR OR PDR WITH INDEX NUMBER
5940  047154  062700  000002                  ADD     #2,R0           ;CHANGE INDEX TO POINT TO NEXT REGISTER
5941  047160  077405                          SOB     R4,1$           ;BRANCH BACK 5 TIMES
5942  047162  012704  000006                  MOV     #6,R4           ;DO NEXT LOOP 6 TIMES ALSO
5943  047166  012737  047200  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
5944  047174  162700  000002          2$:     SUB     #2,R0           ;ADJUST INDEX FOR REGISTER DESIRED
5945  047200  017001  001320          10$:    MOV     @PARTAB(R0),R1  ;READ PAR OR PDR INTO R1
5946  047204  020001                          CMP     R0,R1           ;SEE IF INDEX EQUALS DATA
5947  047206  001403                          BEQ     3$              ;BRANCH IFCORRECT REGISTER WAS READ.
5948  047210  016002  001320                  MOV     PARTAB(R0),R2   ;SAVE ADDRESS OF BAD REGISTER
5949  047214  104036                          ERROR   36              ;DUAL ADDRESSING
5950  047216  077412                  3$:     SOB     R4,2$           ;BRANCH BACK 5 TIMES
5951  047220  012737  047134  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
5952
5953
5954                                  .SBTTL  ************ ENTRY POINT 4 --- STARTING ADDRESS 214 ************
5955                                  .SBTTL  ****************** RELOCATION AND ADDER TESTS ******************
5956                                  ;*
5957                                  ;*      THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER
5958                                  ;*      FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
5959                                  ;*      RELOCATION AND FINALLY WITH 22-BIT RELOCATION.
5960                                  ;*
5961
5962
5963                                  ;;*****************************************************************
5964                                  ;*TEST 46       18-BIT MAPPING ADDER TESTING
5965                                  ;*
5966                                  ;*      THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
5967                                  ;*      FULL ADD PROPERTIES OF THE RELOCATION ADDER.  TWELVE PAIRS
5968                                  ;*      OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES ABOVE
5969                                  ;*      16K.
5970                                  ;*
5971                                  ;;*****************************************************************
5972  047226                          TST46:
5973  047226  000004                          SCOPE
5974  047230  012737  050642  001314          MOV     #TST47,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
5975                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
5976                                  ;
5977                                  ;       THE FOLLOWING CODE WILL CLEAR ALL PAR'S AND PDR'S SO THAT
5978                                  ;       THE RELOCATION ADDERS CAN BE CHECKED, ONLY THE KERNEL I-SPACE
5979                                  ;       PDR'S AND PAR'S WILL BE MAPPED RESIDENT AND READ WRITE.
5980                                  ;
5981  047236                          ENTPT4:
5982  047236  012737  050452  001106          MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
5983  047244  012737  050452  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
5984  047252  012737  000046  001102          MOV     #46,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
5985  047260  013737  001102  177570          MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
```

```
 5986  047266  012705  172300                    MOV     #KIPDR0,R5      ;PUT ADDRESS OF KIPDR0 IN R5
 5987  047272  004737  024156                    JSR     PC,CLRREG       ;CLEAR KERNEL PDR'S
 5988  047276  012705  172340                    MOV     #KIPAR0,R5      ;PUT ADDRESS OF KIPAR0 IN R5
 5989  047302  004737  024156                    JSR     PC,CLRREG       ;CLEAR KERNEL PAR'S
 5990  047306  012705  172200                    MOV     #SIPDR0,R5      ;PUT ADDRESS OF SIPDR0 IN R5
 5991  047312  004737  024156                    JSR     PC,CLRREG       ;CLEAR SUPERVISOR PDR'S
 5992  047316  012705  172240                    MOV     #SIPAR0,R5      ;PUT ADDRESS OF SIPAR0 IN R5
 5993  047322  004737  024156                    JSR     PC,CLRREG       ;CLEAR SUPERVISOR PAR'S
 5994  047326  012705  177600                    MOV     #UIPDR0,R5      ;PUT ADDRESS OF UIPDR0 IN R5
 5995  047332  004737  024156                    JSR     PC,CLRREG       ;CLEAR USER PDR'S
 5996  047336  012705  177640                    MOV     #UIPAR0,R5      ;PUT ADDRESS OF UIPAR0 IN R5
 5997  047342  004737  024156                    JSR     PC,CLRREG       ;CLEAR USER PAR'S
 5998  047346  012700  077406          30$:      MOV     #77406,R0       ;MAKE KERNEL I PAGES 4K, R/W, EXPAND UP
 5999  047352  012702  000010                    MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 6000  047356  012701  172300                    MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 6001  047362  010021                  64$:      MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 6002  047364  077202                            SOB     R2,64$          ;BRANCH BACK TO 64$ IF R2 IS NOT ZERO
 6003  047366  012737  000000  172340            MOV     #000,KIPAR0     ;MAP PAGE 0 TO 0 - 4K
 6004  047374  012737  000200  172342            MOV     #200,KIPAR1     ;MAP PAGE 1 TO 4K - 8K
 6005  047402  012737  000400  172344            MOV     #400,KIPAR2     ;MAP PAGE 2 TO 8K - 12K
 6006  047410  012737  000600  172346            MOV     #600,KIPAR3     ;MAP PAGE 3 TO 12K - 16K
 6007  047416  012737  177600  172356            MOV     #177600,KIPAR7  ;MAP PAGE 7 TO I/O PAGE
 6008  047424  012737  047456  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
 6009  047432  013737  100000  001170            MOV     @#100000,$TMP0  ;SAVE DATA AT TEST LOCATION
 6010  047440  012737  001000  172350            MOV     #1000,@#KIPAR4  ;LOAD PAR4 WITH 1000
 6011  047446  012700  100000                    MOV     #100000,R0      ;PUT VIRTUAL ADDRESS IN R0
 6012  047452  012701  125200                    MOV     #125200,R1      ;PUT DATA PATTERN IN R1
 6013  047456  052737  000400  177572  1$:       BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
 6014  047464  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 6015  047466  010110                            MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 100000
 6016  047470  013702  100000                    MOV     @#100000,R2     ;READ (100000) INTO R2
 6017  047474  000005                            RESET                   ;CLEAR MMR0
 6018  047476  020102                            CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
 6019  047500  001401                            BEQ     2$              ;BRANCH IF DATA MATCHES
 6020  047502  104037                            ERROR   37              ;RELOCATION FAILED
 6021  047504  013737  001170  100000  2$:       MOV     $TMP0,@#100000  ;RESTORE ORIGINAL DATA TO TEST LOCATION
 6022  047512  012737  047544  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
 6023  047520  013737  137776  001170            MOV     @#137776,$TMP0  ;SAVE DATA AT TEST LOCATION
 6024  047526  012737  001252  172350            MOV     #1252,@#KIPAR4  ;LOAD PAR4 WITH 1252
 6025  047534  012700  112576                    MOV     #112576,R0      ;PUT VIRTUAL ADDRESS IN R0
 6026  047540  012701  125201                    MOV     #125201,R1      ;PUT DATA PATTERN IN R1
 6027  047544  052737  000400  177572  3$:       BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
 6028  047552  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 6029  047554  010110                            MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 137776
 6030  047556  013702  137776                    MOV     @#137776,R2     ;READ (137776) INTO R2
 6031  047562  000005                            RESET                   ;CLEAR MMR0
 6032  047564  020102                            CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
 6033  047566  001401                            BEQ     4$              ;BRANCH IF DATA MATCHES
 6034  047570  104037                            ERROR   37              ;RELOCATION FAILED
 6035  047572  013737  001170  137776  4$:       MOV     $TMP0,@#137776  ;RESTORE ORIGINAL DATA TO TEST LOCATION
 6036  047600  012737  047632  001110            MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
 6037  047606  013737  117776  001170            MOV     @#117776,$TMP0  ;SAVE DATA AT TEST LOCATION
 6038  047614  012737  001125  172350            MOV     #1125,@#KIPAR4  ;LOAD PAR4 WITH 1125
 6039  047622  012700  105276                    MOV     #105276,R0      ;PUT VIRTUAL ADDRESS IN R0
 6040  047626  012701  125202                    MOV     #125202,R1      ;PUT DATA PATTERN IN R1
 6041  047632  052737  000400  177572  5$:       BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
```

```
6042  047640  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6043  047642  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 117776
6044  047644  013702  117776                  MOV     @#117776,R2 ;READ (117776) INTO R2
6045  047650  000005                          RESET               ;CLEAR MMR0
6046  047652  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6047  047654  001401                          BEQ     6$          ;BRANCH IF DATA MATCHES
6048  047656  104037                          ERROR   37          ;RELOCATION FAILED
6049  047660  013737  001170  117776  6$:     MOV     $TMP0,@#117776 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6050  047666  012737  047720  001110          MOV     #7$,$LPERR  ;SET LOOP ON ERROR POINTER TO 7$
6051  047674  013737  102000  001170          MOV     @#102000,$TMP0 ;SAVE DATA AT TEST LOCATION
6052  047702  012737  001010  172350          MOV     #1010,@#KIPAR4 ;LOAD PAR4 WITH 1010
6053  047710  012700  101000                  MOV     #101000,R0  ;PUT VIRTUAL ADDRESS IN R0
6054  047714  012701  125203                  MOV     #125203,R1  ;PUT DATA PATTERN IN R1
6055  047720  052737  000400  177572  7$:     BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6056  047726  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6057  047730  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 102000
6058  047732  013702  102000                  MOV     @#102000,R2 ;READ (102000) INTO R2
6059  047736  000005                          RESET               ;CLEAR MMR0
6060  047740  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6061  047742  001401                          BEQ     8$          ;BRANCH IF DATA MATCHES
6062  047744  104037                          ERROR   37          ;RELOCATION FAILED
6063  047746  013737  001170  102000  8$:     MOV     $TMP0,@#102000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6064  047754  012737  050006  001110          MOV     #9$,$LPERR  ;SET LOOP ON ERROR POINTER TO 9$
6065  047762  013737  142000  001170          MOV     @#142000,$TMP0 ;SAVE DATA AT TEST LOCATION
6066  047770  012737  001314  172350          MOV     #1314,@#KIPAR4 ;LOAD PAR4 WITH 1314
6067  047776  012700  110400                  MOV     #110400,R0  ;PUT VIRTUAL ADDRESS IN R0
6068  050002  012701  125204                  MOV     #125204,R1  ;PUT DATA PATTERN IN R1
6069  050006  052737  000400  177572  9$:     BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6070  050014  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6071  050016  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 142000
6072  050020  013702  142000                  MOV     @#142000,R2 ;READ (142000) INTO R2
6073  050024  000005                          RESET               ;CLEAR MMR0
6074  050026  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6075  050030  001401                          BEQ     10$         ;BRANCH IF DATA MATCHES
6076  050032  104037                          ERROR   37          ;RELOCATION FAILED
6077  050034  013737  001170  142000  10$:    MOV     $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6078  050042  012737  050074  001110          MOV     #11$,$LPERR ;SET LOOP ON ERROR POINTER TO 11$
6079  050050  013737  122000  001170          MOV     @#122000,$TMP0 ;SAVE DATA AT TEST LOCATION
6080  050056  012737  001104  172350          MOV     #1104,@#KIPAR4 ;LOAD PAR4 WITH 1104
6081  050064  012700  111400                  MOV     #111400,R0  ;PUT VIRTUAL ADDRESS IN R0
6082  050070  012701  125205                  MOV     #125205,R1  ;PUT DATA PATTERN IN R1
6083  050074  052737  000400  177572  11$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6084  050102  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6085  050104  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 122000
6086  050106  013702  122000                  MOV     @#122000,R2 ;READ (122000) INTO R2
6087  050112  000005                          RESET               ;CLEAR MMR0
6088  050114  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6089  050116  001401                          BEQ     12$         ;BRANCH IF DATA MATCHES
6090  050120  104037                          ERROR   37          ;RELOCATION FAILED
6091  050122  013737  001170  122000  12$:    MOV     $TMP0,@#122000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6092  050130  012737  050162  001110          MOV     #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
6093  050136  013737  142000  001170          MOV     @#142000,$TMP0 ;SAVE DATA AT TEST LOCATION
6094  050144  012737  001356  172350          MOV     #1356,@#KIPAR4 ;LOAD PAR4 WITH 1356
6095  050152  012700  104200                  MOV     #104200,R0  ;PUT VIRTUAL ADDRESS IN R0
6096  050156  012701  125206                  MOV     #125206,R1  ;PUT DATA PATTERN IN R1
6097  050162  052737  000400  177572  13$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
```

```
6098  050170  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6099  050172  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 142000
6100  050174  013702  142000                  MOV     @#142000,R2 ;READ (142000) INTO R2
6101  050200  000005                          RESET               ;CLEAR MMR0
6102  050202  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6103  050204  001401                          BEQ     14$         ;BRANCH IF DATA MATCHES
6104  050206  104037                          ERROR   37          ;RELOCATION FAILED
6105  050210  013737  001170  142000  14$:    MOV     $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6106  050216  012737  050250  001110          MOV     #15$,$LPERR ;SET LOOP ON ERROR POINTER TO 15$
6107  050224  013737  142000  001170          MOV     @#142000,$TMP0 ;SAVE DATA AT TEST LOCATION
6108  050232  012737  001242  172350          MOV     #1242,@#KIPAR4 ;LOAD PAR4 WITH 1242
6109  050240  012700  115600                  MOV     #115600,R0  ;PUT VIRTUAL ADDRESS IN R0
6110  050244  012701  125207                  MOV     #125207,R1  ;PUT DATA PATTERN IN R1
6111  050250  052737  000400  177572  15$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6112  050256  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6113  050260  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 142000
6114  050262  013702  142000                  MOV     @#142000,R2 ;READ (142000) INTO R2
6115  050266  000005                          RESET               ;CLEAR MMR0
6116  050270  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6117  050272  001401                          BEQ     16$         ;BRANCH IF DATA MATCHES
6118  050274  104037                          ERROR   37          ;RELOCATION FAILED
6119  050276  013737  001170  142000  16$:    MOV     $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6120  050304  012737  050336  001110          MOV     #17$,$LPERR ;SET LOOP ON ERROR POINTER TO 17$
6121  050312  013737  142000  001170          MOV     @#142000,$TMP0 ;SAVE DATA AT TEST LOCATION
6122  050320  012737  001377  172350          MOV     #1377,@#KIPAR4 ;LOAD PAR4 WITH 1377
6123  050326  012700  102100                  MOV     #102100,R0  ;PUT VIRTUAL ADDRESS IN R0
6124  050332  012701  125210                  MOV     #125210,R1  ;PUT DATA PATTERN IN R1
6125  050336  052737  000400  177572  17$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6126  050344  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6127  050346  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 142000
6128  050350  013702  142000                  MOV     @#142000,R2 ;READ (142000) INTO R2
6129  050354  000005                          RESET               ;CLEAR MMR0
6130  050356  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6131  050360  001401                          BEQ     18$         ;BRANCH IF DATA MATCHES
6132  050362  104037                          ERROR   37          ;RELOCATION FAILED
6133  050364  013737  001170  142000  18$:    MOV     $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6134  050372  012737  050424  001110          MOV     #19$,$LPERR ;SET LOOP ON ERROR POINTER TO 19$
6135  050400  013737  142000  001170          MOV     @#142000,$TMP0 ;SAVE DATA AT TEST LOCATION
6136  050406  012737  001221  172350          MOV     #1221,@#KIPAR4 ;LOAD PAR4 WITH 1221
6137  050414  012700  117700                  MOV     #117700,R0  ;PUT VIRTUAL ADDRESS IN R0
6138  050420  012701  125211                  MOV     #125211,R1  ;PUT DATA PATTERN IN R1
6139  050424  052737  000400  177572  19$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
6140  050432  000240                          NOP                 ;THIS IS A SYNC POINT FOR SCOPING
6141  050434  010110                          MOV     R1,(R0)     ;TRY TO LOAD DATA PATTERN INTO 142000
6142  050436  013702  142000                  MOV     @#142000,R2 ;READ (142000) INTO R2
6143  050442  000005                          RESET               ;CLEAR MMR0
6144  050444  020102                          CMP     R1,R2       ;SEE IF DATA MATCHES PATTERN
6145  050446  001401                          BEQ     20$         ;BRANCH IF DATA MATCHES
6146  050450  104037                          ERROR   37          ;RELOCATION FAILED
6147  050452  013737  001170  142000  20$:    MOV     $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6148  050460  012737  050512  001110          MOV     #21$,$LPERR ;SET LOOP ON ERROR POINTER TO 21$
6149  050466  013737  140000  001170          MOV     @#140000,$TMP0 ;SAVE DATA AT TEST LOCATION
6150  050474  012737  001377  172350          MOV     #1377,@#KIPAR4 ;LOAD PAR4 WITH 1377
6151  050502  012700  100100                  MOV     #100100,R0  ;PUT VIRTUAL ADDRESS IN R0
6152  050506  012701  125212                  MOV     #125212,R1  ;PUT DATA PATTERN IN R1
6153  050512  052737  000400  177572  21$:    BIS     #BIT8,@#MMR0 ;TURN ON DESTINATION ONLY RELOCATION
```

```
6154  050520  000240                          NOP             ;THIS IS A SYNC POINT FOR SCOPING
6155  050522  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 140000
6156  050524  013702  140000                  MOV     @#140000,R2     ;READ (140000) INTO R2
6157  050530  000005                          RESET                   ;CLEAR MMR0
6158  050532  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
6159  050534  001401                          BEQ     22$             ;BRANCH IF DATA MATCHES
6160  050536  104037                          ERROR   37              ;RELOCATION FAILED
6161  050540  013737  001170  140000  22$:    MOV     $TMP0,@#140000  ;RESTORE ORIGINAL DATA TO TEST LOCATION
6162  050546  012737  050600  001110          MOV     #23$,$LPERR     ;SET LOOP ON ERROR POINTER TO 23$
6163  050554  013737  140000  001170          MOV     @#140000,$TMP0  ;SAVE DATA AT TEST LOCATION
6164  050562  012737  001370  172350          MOV     #1370,@#KIPAR4  ;LOAD PAR4 WITH 1370
6165  050570  012700  101000                  MOV     #101000,R0      ;PUT VIRTUAL ADDRESS IN R0
6166  050574  012701  125213                  MOV     #125213,R1      ;PUT DATA PATTERN IN R1
6167  050600  052737  000400  177572  23$:    BIS     #BIT8,@#MMR0    ;TURN ON DESTINATION ONLY RELOCATION
6168  050606  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6169  050610  010110                          MOV     R1,(R0)         ;TRY TO LOAD DATA PATTERN INTO 140000
6170  050612  013702  140000                  MOV     @#140000,R2     ;READ (140000) INTO R2
6171  050616  000005                          RESET                   ;CLEAR MMR0
6172  050620  020102                          CMP     R1,R2           ;SEE IF DATA MATCHES PATTERN
6173  050622  001401                          BEQ     24$             ;BRANCH IF DATA MATCHES
6174  050624  104037                          ERROR   37              ;RELOCATION FAILED
6175  050626  013737  001170  140000  24$:    MOV     $TMP0,@#140000  ;RESTORE ORIGINAL DATA TO TEST LOCATION
6176  050634  012737  047346  001110          MOV     #30$,$LPERR     ;SET LOOP ON ERROR POINTER TO START OF TEST
6177
6178                                  ;;*****************************************************************
6179                                  ;*TEST 47        18-BIT MAPPING CARRY PROPAGATION
6180                                  ;*
6181                                  ;*      THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY
6182                                  ;*      PROPAGATION OF THE RELOCATION ADDER.  SINCE THIS TEST SCANS
6183                                  ;*      MEMORY FROM 00100000 TO 00750000 ON 2K BOUNDARIES, IT WILL
6184                                  ;*      REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE SIZE
6185                                  ;*      JUMPERS.  THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE
6186                                  ;*      THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER THE
6187                                  ;*      HOLE.
6188                                  ;;*****************************************************************
6189  050642                          TST47:
6190  050642  000004                          SCOPE
6191  050644  012737  051436  001314          MOV     #TST50,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
6192                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
6193  050652  005037  001302          20$:    CLR     HOLFLG          ;MAKE SURE HOLE FLAG STARTS AT 0
6194  050656  012737  051272  000004          MOV     #10$,ERRVEC     ;SET ERRVEC POINTER TO 10$
6195  050664  012737  000777  172350          MOV     #777,KIPAR4     ;LOAD PAR4 WITH STARTING BASE
6196  050672  012737  001000  172352          MOV     #1000,KIPAR5    ;START ADDRESSING WITH 16K
6197  050700  012700  100100                  MOV     #100100,R0      ;LOAD VIRTUAL ADDR FOR PAR4 IN R0
6198  050704  012701  120000                  MOV     #120000,R1      ;LOAD VIRTUAL ADDR FOR PAR5 IN R1
6199  050710  012702  001000                  MOV     #1000,R2        ;LOAD DATA PATTERN INTO R2
6200  050714  012737  051004  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
6201  050722  012737  000001  177572          MOV     #1,@#MMR0       ;TURN ON 18-BIT MAPPING
6202                                          ;
6203                                  ;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM
6204                                          ;
6205  050730  012737  051004  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
6206  050736  005037  001256                  CLR     PCPUER          ;CLEAR CPU ERROR FLAG
6207  050742  011137  001170                  MOV     (R1),$TMP0      ;SAVE DATA AT TEST LOCATION
6208  050746  005737  001256                  TST     PCPUER          ;SEE IF THERE WAS A CPU TRAP
6209  050752  001014                          BNE     2$              ;BRANCH IF TRAP OCCURRED
```

```
                                                TST     HOLFLG          ;SEE IF A HOLE WAS FOUND IN MEMORY
                                                BEQ     2$              ;BRANCH IF NO HOLE WAS FOUND
                                                MOV     KIPAR5,$TMP2    ;SAVE PAR THAT POINTS TO END OF HOLE
                                                MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
                                                ERROR   40              ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
                                                CLR     HOLFLG          ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
                                        2$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
                                                MOV     R2,(R0)         ;LOAD TEST PATTERN INTO TEST LOCATION
                                                MOV     (R1),R3         ;READ TEST LOCATION VIA DIFFERENT V. A.
                                                CMP     R2,R3           ;SEE IF CORRECT LOCATION WAS REFERENCED
                                                BEQ     3$              ;BRANCH IF CORRECT DATA WAS OBTAINED
                                                ERROR   42              ;BAD RELOCATION
                                        3$:     MOV     $TMP0,(R1)      ;RESTORE ORIGINAL DATA TO TEST LOCATION
                                                ADD     #100,KIPAR4     ;CHANGE BASE ADDRESS
                                                ADD     #100,KIPAR5     ;CHANGE BASE ADDRESS
                                                INC     R2              ;CHANGE DATA PATTERN
                                                CMP     #7500,KIPAR5    ;SEE IF PAST LAST ADDRESS
                                                BHIS    1$              ;BRANCH IF NOT PAST LAST ADDRESS
                                                TST     HOLFLG          ;SEE IF YOU ARE IN THE MIDDLE OF A HOLE
                                                BEQ     4$              ;BRANCH IF NOT IN MIDDLE OF A HOLE
                                                ERROR   41              ;IN MIDDLE OF A HOLE IN MEMORY
                                                                        ;HOLFLG HAS NO. OF TIME OUTS $TMP1

                                                                        ;HAS PAR OF FIRST TIMEOUT
                                        .SBTTL          TEST 'SAPN UNIBUS ADRS L' IN 18-BIT MAPPING
                                        ;*
                                        ;*      ADDRESS 760000 IS GENERATED BY CARRY PROPAGATION WHILE THE
                                        ;*      PROGRAM IS EXPECTING A TIME OUT.  (760000 IS THE FIRST I/O
                                        ;*      PAGE ADDRESS AND IS NOT IMPLEMENTED EXCEPT FOR DIAGNOSTICS)
                                        ;*      KIPAR4 WILL CONTAIN 007577 AND R0 HAS 100100 (WHICH
                                        ;*      SELECTS KIPAR4 AND GENERATES ADDRESS 760000).
                                        ;*      IF THE WRONG BIT IN THE CPU ERROR REGISTER IS SET OR IF
                                        ;*      NO TRAP TO 'ERRVEC' OCCURS AN ERROR IS REPORTED.
                                        ;*

 051062  012737  024516  000004 4$:     MOV     #CPUER,ERRVEC   ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
 051070  012737  051122  001110         MOV     #40$,$LPERR     ;SET LOOP ON ERROR POINTER TO 40$
 051076  012737  007577  172350         MOV     #7577,KIPAR4    ;LOAD KIPAR4 WITH 007577
 051104  012702  007600                 MOV     #7600,R2        ;LOAD DATA PATTERN INTO R2
 051110  012737  000020  001222         MOV     #20,CPUEXP      ;EXPECTING UNIBUS TIMEOUT
 051116  005037  001256                 CLR     PCPUER          ;CLEAR CPU TRAP FLAG
 051122  000240                 40$:    NOP                     ;THIS A IS SYNC POINT FOR SCOPING
 051124  010210                         MOV     R2,(R0)         ;LOAD 760000 THRU PAGE 4
                                                                 ;THIS INSTRUCTION SHOULD TIME OUT
                                                                 ;OVER THE UNIBUS.
 051126  005737  001256                 TST     PCPUER          ;SEE IF TRAP OCCURRED
 051132  001001                         BNE     6$              ;BRANCH IF TRAP
 051134  104043                         ERROR   43              ;NO CPU TRAP

                                        .SBTTL          TEST 'SAPN NOT CACHE ADRS H' 18-BIT MAPPING
                                        ;*
                                        ;*      ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL
                                        ;*      CAUSE '18BIT WRAPAROUND' TO BE ASSERTED, KNOCKING DOWN
                                        ;*      'NOT CACHE ADRS'.  THEN, IF THERE IS LESS THAN 120K OF MEMORY
                                        ;*      ON THE SYSTEM, THE SIZE REGISTER IS USED AS A PAR AND A
                                        ;*      CARRY IS PROPAGATED TO CAUSE '18BIT OVERFLOW' TO BE ASSERTED
```

```
                                        ;*      WHICH SHOULD GENERATE 'NOT CACHE ADRS'.
                                        ;*
051136  012737  051162  001110  6$:     MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
051144  005037  001222                  CLR     CPUEXP          ;NO TRAPS THRU ERRVEC EXPECTED HERE
051150  012737  007777  172350          MOV     #7777,KIPAR4    ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
051156  005037  000000                  CLR     @#000000        ;CLEAR ADDRESS ZERO
051162  000240                  16$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
051164  013701  100100                  MOV     @#100100,R1     ;THIS SHOULD READ ADDRESS ZERO INTO R1
051170  005701                          TST     R1              ;SEE IF YOU REALLY READ ADDRESS ZERO
051172  001401                          BEQ     7$              ;BRANCH IF YOU READ ADDRESS ZERO
051174  104044                          ERROR   44              ;DIDN'T READ ADDRESS ZERO
051176  012737  051234  001110  7$:     MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
051204  022737  007377  177760          CMP     #7377,SIZELO    ;IS SIZE REGISTER L.E. THAN 7377
051212  101421                          BLOS    8$              ;BRANCH IF MORE THAN 120K ON SYSTEM
051214  012737  000040  001222          MOV     #40,CPUEXP      ;EXPECTING CACHE NON-EXISTANT MEMORY
051222  005037  001256                  CLR     PCPUER          ;CLEAR TRAP THRU ERRVEC FLAG
051226  013737  177760  172350          MOV     SIZELO,KIPAR4   ;GET READY TO GENERATE NON-EXIST ADDR.
051234  000240                  17$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
051236  013701  100100                  MOV     @#100100,R1     ;READ FROM NON-EXISTANT ADDRESS
051242  005737  001256                  TST     PCPUER          ;SEE IF TRAP THRU ERRVEC OCCURRED
051246  001003                          BNE     8$              ;BRANCH TO EXIT IF TRAP HAPPENED
051250  012700  100100                  MOV     #100100,R0      ;SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
051254  104045                          ERROR   45              ;NO TRAP THRU ERRVEC
051256  005037  001222          8$:     CLR     CPUEXP          ;NO CPU TRAPS EXPECTED
051262  012737  050652  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
051270  000462                          BR      TST50           ;;BRANCH TO NEXT TEST

                                        ;;******************  TRAP TO HERE THRU ERRVEC  **********************

051272  012637  001304          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
051276  012637  001306                  MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
051302  013737  177766  001256          MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
051310  022737  000040  001256          CMP     #40,PCPUER      ;WAS TRAP NON-EXISTANT MEMORY
051316  001012                          BNE     12$             ;BRANCH IF MEMORY EXISTS
051320  023737  172350  177760          CMP     KIPAR4,SIZELO   ;SEE IF PAR 4 MATCHES SIZE REGISTER
051326  001004                          BNE     11$             ;BRANCH IF NO MATCH, POSSIBLE ERROR
                                                                ;IN COMPARE CIRCUIT
051330  012737  051062  001304          MOV     #4$,OLDPC       ;CHANGE RETURN ADDRESS IF AT TOP OF MEM
051336  000430                          BR      14$             ;BRANCH TO EXIT
051340  104050                  11$:    ERROR   50              ;COMPARE CIRCUIT FOR SIZE JUMPERS BAD
051342  000426                          BR      14$             ;BRANCH TO EXIT & CONTINUE
                                        ;;
                                        ;;********** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
                                        ;;*********  THERE MIGHT BE A HOLE IN MEMORY.
                                        ;;
051344  022737  050652  001256  12$:    CMP     #20$,PCPUER     ;SEE IF ADDRESS TIMED OUT
051352  001011                          BNE     13$             ;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
051354  005737  001302                  TST     HOLFLG          ;HAS THIS HAPPENED BEFORE?
051360  001003                          BNE     15$             ;BRANCH IF NOT FIRST TIMEOUT
051362  013737  172352  001172          MOV     KIPAR5,$TMP1    ;SAVE PAR WHEN HOLE FIRST DISCOVERED
051370  005237  001302          15$:    INC     HOLFLG          ;KEEP COUNT OF CONSECUTIVE TIMEOUTS
051374  000411                          BR      14$             ;BRANCH TO EXIT AND CONTINUE TEST
051376  013737  001304  001260  13$:    MOV     OLDPC,BADPC     ;MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
051404  104002                          ERROR   2               ;UNEXPECTED TRAP THRU ERRVEC
051406  013737  001106  001304          MOV     $LPADR,OLDPC    ;RETURN TO START OF TEST
```

```
                                        CLR     HOLFLG          ;CLEAR HOLE FLAG IN CASE IT WAS SET
                                14$:    CLR     CPUERR          ;CLEAR THE CPU ERROR REGISTER
                                        MOV     OLDPS,-(KSP)    ;PUSH OLD PROCESSOR STATUS ON STACK
                                        MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
                                        RTI                     ;RETURN TO TEST AND CONTINUE


                                ;;*****************************************************************
                                ;*TEST 50        22-BIT MAPPING CARRY PROPAGATION
                                ;*
                                ;*      THIS TEST USES FULL 22-BIT RELOCATION TO CHECK THE CARRY
                                ;*      PROPAGATION THAT PERTAINS TO 22 BIT ADDRESSES.  THIS TEST
                                ;*      ALSO SCANS MEMORY THIS TIME FROM ADDRESS 00740000
                                ;*      TO 16740000 OR THE SIZE JUMPERS, ON 8K BOUNDARIES.  AGAIN
                                ;*      IF ANY HOLES ARE FOUND THE ADDRESS WHERE THEY ARE DISCOVERED
                                ;*      AND THE FIRST GOOD ADDRESS AFTER THE HOLE WILL BE REPORTED
                                ;*
                                ;;*****************************************************************
051436                          TST50:
051436  000004                          SCOPE
051440  012737  052256  001314        MOV     #TST51,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                ;
                                ;       22-BIT MAPPING STARTS HERE AND CONTINUES FOR THE REST OF
                                ;       THE PROGRAM.
                                ;
051446  022737  007377  177760          CMP     #7377,SIZELO    ;IS THERE AT LEAST 120K ON SYSTEM?
051454  003102                          BGT     4$              ;BRANCH IF LESS THAN 120K
051456  012737  000020  172516          MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
051464  012737  052112  000004  20$:    MOV     #10$,ERRVEC     ;SET ERRVEC POINTER TO 10$
051472  005037  001302                  CLR     HOLFLG          ;START HOLE FLAG AT ZERO
051476  012700  100100                  MOV     #100100,R0      ;LOAD VIRTUAL ADDRESS FOR PAGE 4
051502  012701  120000                  MOV     #120000,R1      ;LOAD VIRTUAL ADDRESS FOR PAGE 5
051506  012737  007377  172350          MOV     #7377,KIPAR4    ;LOAD BASE ADDRESS INTO PAR 4
051514  012737  007400  172352          MOV     #7400,KIPAR5    ;LOAD BASE ADDRESS +100 INTO PAR5
051522  012702  007400                  MOV     #7400,R2        ;LOAD DATA PATTERN INTO R2
051526  012737  051602  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
051534  005037  001256                  CLR     PCPUER          ;CLEAR CPU TRAP FLAG
051540  011137  001170                  MOV     (R1),$TMP0      ;SAVE DATA AT TEST LOCATION USING PAGE 5
051544  005737  001256                  TST     PCPUER          ;SEE IF CPU TRAP OCCURRED
051550  001014                          BNE     2$              ;BRANCH IF TRAP OCCURED
051552  005737  001302                  TST     HOLFLG          ;SEE IF A HOLE IN MEMORY WAS FOUND
051556  001411                          BEQ     2$              ;BRANCH IF NO HOLE WAS FOUND
051560  013737  172352  001174          MOV     KIPAR5,$TMP2    ;SAVE PAR THAT POINTS TO END OF HOLE
051566  012737  051464  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
051574  104125                          ERROR   125             ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
051576  005037  001302                  CLR     HOLFLG          ;CLEAR FLAG IN CASE OF MORE HOLES
051602  000240                  2$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
051604  010210                          MOV     R2,(R0)         ;WRITE DATA PATTERN INTO TEST LOCATION
051606  011103                          MOV     (R1),R3         ;READ TEST LOCATION  VIA DIFFERENT VIRT.ADDR
051610  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
051612  001401                          BEQ     3$              ;BRANCH IF DATA IS GOOD
051614  104046                          ERROR   46              ;BAD RELOCATION 22-BIT MAPPING
051616  013711  001170          3$:     MOV     $TMP0,(R1)      ;RESTORE ORIGINAL DATA USING PAGE 5
051622  062702  000400                  ADD     #400,R2         ;CHANGE DATA PATTERN
```

```
6378  051626  062737  000400  172350          ADD     #400,KIPAR4     ;GET READY TO TEST NEXT BIT
6379  051634  062737  000400  172352          ADD     #400,KIPAR5     ;NEW PHYSICAL ADDRESS
6380  051642  022737  167400  172352          CMP     #167400,KIPAR5  ;MAKE SURE YOU DON'T GET ON THE UNIBUS
6381  051650  103326                          BHIS    1$              ;BRANCH IF PAR 5 IS NOT PAST LIMIT
6382  051652  005737  001302                  TST     HOLFLG          ;SEE IF MEMORY ENDS WITH A HOLE
6383  051656  001401                          BEQ     4$              ;BRANCH IF NO HOLE AT END OF MEMORY
6384  051660  104126                          ERROR   126             ;HOLE AT END OF MEMORY
6385
6386                                  .SBTTL          TEST 'SAPN UNIBUS ADRS L' IN 22-BIT MAPPING
6387                                  ;*
6388                                  ;*      UNIBUS ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION
6389                                  ;*      SINCE THE MAP IS DISABLED THIS SHOULD REFERENCE PHYSICAL
6390                                  ;*      ADDRESS 000000.
6391                                  ;*
6392
6393  051662  012737  000020  172516  4$:     MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
6394  051670  012737  024516  000004          MOV     #CPUER,ERRVEC   ;RESTORE NORMAL CPU TRAP ROUTINE
6395  051676  012737  000020  001222          MOV     #20,CPUEXP      ;POSSIBLE U.B. TIME OUT
6396  051704  012737  167777  172350          MOV     #167777,KIPAR4  ;GET READY TO TEST U.B. ADDRESS 0
6397  051712  012737  000000  172352          MOV     #0,KIPAR5       ;SHOULD GO TO PHYSICAL ADDRESS 0
6398  051720  012702  017000                  MOV     #17000,R2       ;LOAD DATA PATTERN INTO R2
6399  051724  012737  051736  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
6400  051732  011037  001170                  MOV     (R0),$TMP0      ;SAVE DATA IN LOCATION 0 USING PAGE 4
6401  051736  000240                  5$:     NOP                     ;THIS A IS SYNC POINT FOR SCOPING
6402  051740  010210                          MOV     R2,(R0)         ;LOAD DATA PATTERN INTO TEST LOCATION
6403  051742  011103                          MOV     (R1),R3         ;READ TEST LOCATION VIA DIFFERENT V. A.
6404  051744  013710  001170                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL DATA USING PAGE 4
6405  051750  020203                          CMP     R2,R3           ;SEE IF DATA MATCHES
6406  051752  001401                          BEQ     6$              ;BRANCH IF DATA IS GOOD
6407  051754  104047                          ERROR   47              ;BAD RELOCATION, UNIBUS ADDRESS
6408
6409                                  .SBTTL          TEST 'SAPN NOT CACHE ADRS H' 22-BIT MAPPING
6410                                  ;*
6411                                  ;*      ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL
6412                                  ;*      CAUSE '22BIT WRAPAROUND' TO BE ASSERTED, KNOCKING DOWN
6413                                  ;*      'NOT CACHE ADRS'.  THEN THE SIZE REGISTER IS USED AS A PAR AND A
6414                                  ;*      CARRY IS PROPAGATED TO CAUSE 'ADRS OVERFLOW' TO BE ASSERTED
6415                                  ;*      WHICH SHOULD GENERATE 'NOT CACHE ADRS'.
6416                                  ;*
6417
6418  051756  012737  052002  001110  6$:     MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
6419  051764  005037  001222                  CLR     CPUEXP          ;NO TRAPS THRU ERRVEC EXPECTED HERE
6420  051770  012737  177777  172350          MOV     #177777,KIPAR4  ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
6421  051776  005037  000000                  CLR     @#000000        ;CLEAR ADDRESS ZERO
6422  052002  000240                  16$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6423  052004  013701  100100                  MOV     @#100100,R1     ;THIS SHOULD READ ADDRESS ZERO INTO R1
6424  052010  005701                          TST     R1              ;SEE IF YOU REALLY READ ADDRESS ZERO
6425  052012  001401                          BEQ     7$              ;BRANCH IF YOU READ ADDRESS ZERO
6426  052014  104044                          ERROR   44              ;DIDN'T READ ADDRESS ZERO
6427  052016  012737  052054  001110  7$:     MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
6428  052024  022737  167777  177760          CMP     #167777,SIZELO  ;IS SIZE REGISTER L.E. TO 167777
6429  052032  101421                          BLOS    8$              ;BRANCH IF MAXIMUM MEMORY IS ON SYSTEM
6430  052034  012737  000040  001222          MOV     #40,CPUEXP      ;EXPECTING CACHE NON-EXISTANT MEMORY
6431  052042  005037  001256                  CLR     PCPUER          ;CLEAR TRAP THRU ERRVEC FLAG
6432  052046  013737  177760  172350          MOV     SIZELO,KIPAR4   ;GET READY TO GENERATE NON-EXIST ADDR.
6433  052054  000240                  17$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
```

```
6434  052056  013701  100100          MOV     @#100100,R1     ;READ FROM NON-EXISTANT ADDRESS
6435  052062  005737  001256          TST     PCPUER          ;SEE IF TRAP THRU ERRVEC OCCURRED
6436  052066  001003                  BNE     8$              ;BRANCH TO EXIT IF TRAP HAPPENED
6437  052070  012700  100100          MOV     #100100,R0      ;SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
6438  052074  104045                  ERROR   45              ;NO TRAP THRU ERRVEC
6439  052076  005037  001222  8$:     CLR     CPUEXP          ;NO CPU TRAPS EXPECTED
6440  052102  012737  051464  001110  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
6441  052110  000462                  BR      TST51           ;;BRANCH TO NEXT TEST
6442
6443
6444                          ;;**************************  TRAP TO HERE THRU ERRVEC ******************
6445
6446  052112  012637  001304  10$:     MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
6447  052116  012637  001306          MOV     (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
6448  052122  013737  177766  001256  MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER FOR TYPING
6449  052130  022737  000040  001256  CMP     #40,PCPUER      ;WAS TRAP NON-EXISTANT MEMORY
6450  052136  001012                  BNE     12$             ;BRANCH IF MEMORY EXISTS
6451  052140  023737  172350  177760  CMP     KIPAR4,SIZELO   ;SEE IF PAR 4 MATCHES SIZE REGISTER
6452  052146  001004                  BNE     11$             ;BRANCH IF NO MATCH, POSSIBLE ERROR
6453                                                          ;IN COMPARE CIRCUIT
6454  052150  012737  051662  001304  MOV     #4$,OLDPC       ;CHANGE RETURN ADDRESS IF AT TOP OF MEM
6455  052156  000430                  BR      14$             ;BRANCH TO EXIT
6456  052160  104050          11$:    ERROR   50              ;COMPARE CIRCUIT FOR SIZE JUMPERS BAD
6457  052162  000426                  BR      14$             ;BRANCH TO EXIT & CONTINUE
6458                          ;;
6459                          ;;********** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
6460                          ;;********** THERE MIGHT BE A HOLE IN MEMORY.
6461                          ;;
6462  052164  022737  051464  001256  12$:    CMP     #20$,PCPUER     ;SEE IF ADDRESS TIMED OUT
6463  052172  001011                  BNE     13$             ;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
6464  052174  005737  001302          TST     HOLFLG          ;HAS THIS HAPPENED BEFORE?
6465  052200  001003                  BNE     15$             ;BRANCH IF NOT FIRST TIMEOUT
6466  052202  013737  172352  001172  MOV     KIPAR5,$TMP1    ;SAVE PAR WHEN HOLE FIRST DISCOVERED
6467  052210  005237  001302  15$:    INC     HOLFLG          ;KEEP COUNT OF CONSECUTIVE TIMEOUTS
6468  052214  000411                  BR      14$             ;BRANCH TO EXIT AND CONTINUE TEST
6469  052216  013737  001304  001260  13$:    MOV     OLDPC,BADPC     ;MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
6470  052224  104002                  ERROR   2               ;UNEXPECTED TRAP THRU ERRVEC
6471  052226  013737  001106  001304  MOV     $LPADR,OLDPC    ;RETURN TO START OF TEST
6472  052234  005037  001302          CLR     HOLFLG          ;CLEAR HOLE FLAG IN CASE IT WAS SET
6473  052240  005037  177766  14$:    CLR     CPUERR          ;CLEAR THE CPU ERROR REGISTER
6474  052244  013746  001306          MOV     OLDPS,-(KSP)    ;PUSH OLD PROCESSOR STATUS ON STACK
6475  052250  013746  001304          MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
6476  052254  000002                  RTI                     ;RETURN TO TEST AND CONTINUE
6477
6478
6479
6480                          .SBTTL  ************ ENTRY POINT 5 --- STARTING ADDRESS 220 ************
6481                          .SBTTL  ******** MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS ********
6482                          ;*
6483                          ;*      THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT
6484                          ;*      AND TRAP LOGIC ON PAGES 'SAPL', 'SSRC', AND 'SSRD'.  IT WILL
6485                          ;*      ALSO CHECK OUT BITS <07:01> OF MMR0, AND ALL BITS OF MMR2.
6486                          ;*      IT THEN CHECKS THAT KERNEL MODE IS ALWAYS CLOCKED DURING
6487                          ;*      A TRAP SEQUENCE SO THAT THE VECTOR IS PICKED UP FROM KERNEL
6488                          ;*      SPACE.
6489
```

```
                                ;;*****************************************************************
                                ;*TEST 51       PAGE LENGTH FAULTS - UPWARD EXPANSION
                                ;*
                                ;*      THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS ON PAGE 'SAPL'
                                ;*      AND THE LOGIC THAT GENERATES 'SAPL LENGTH FAULT'.  IT TRIES
                                ;*      EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS.  THE
                                ;*      TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
                                ;*      OCCURS.  IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
                                ;*      THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
                                ;*
                                ;;*****************************************************************
052256                          TST51:
                                        SCOPE
052256  000004
052260  012737  052610  001314        MOV     #TST52,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
052266  012737  000024  001204        MOV     #24,$TIMES      ;;DO 24 ITERATIONS
052274                          ENTPT5:
052274  012737  052434  001106        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
052302  012737  052434  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
052310  012737  000051  001102        MOV     #51,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
052316  013737  001102  177570        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
052324  012737  077406  172300        MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
052332  012737  077406  172302        MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
052340  012737  077406  172304        MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
052346  012737  077406  172306        MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
052354  012737  077406  172316        MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
052362  012737  000000  172340        MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
052370  012737  000200  172342        MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
052376  012737  000400  172344        MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
052404  012737  000600  172346        MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
052412  012737  177600  172356        MOV     #177600,KIPAR7  ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
052420  012737  000001  177572        MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
052426  012737  000020  172516        MOV     #BIT4,MMR3      ;ENABLE 22-BIT RELOCATION IF NOT ON
052434  012737  000006  172310  20$:   MOV     #000006,@#KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
052442  012737  001000  172350        MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
052450  012700  172311                MOV     #KIPDR4+1,R0    ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
052454  012737  052506  001110        MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
052462  005037  001246          1$:   CLR     PMMR0           ;CLEAR LOCATION THAT HOLDS MMR0
052466  012702  100000                MOV     #100000,R2      ;PUT VIRTUAL ADDRESS INTO R2
052472  010203                  2$:   MOV     R2,R3           ;PUT VIRTUAL ADDRESS INTO R3 TOO
052474  072327  177772                ASH     #-6,R3          ;RIGHT SHIFT 6 BITS SO IT WILL
                                                                ;MATCH THE PLF
052500  042703  177600                BIC     #177600,R3      ;CLEAR BITS THAT ARE SET IN UPPER BYTE
052504  111001                        MOVB    (R0),R1         ;SAVE PLF IN R1 FOR LATER COMPARISON
052506  012737  040011  001224  3$:   MOV     #040011,MMEXP   ;POTENTIAL ABORT CONDITION: PAGE LENGTH
                                                                ;ERROR KERNEL, I-SPACE, PAGE 4
052514  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
052516  011204                        MOV     (R2),R4         ;READ USING V.A. IN R2
052520  005037  001224                CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION
052524  005737  001246                TST     PMMR0           ;SEE IF ABORT OCCURRED YET
052530  001405                        BEQ     4$              ;BRANCH IF NO ABORT YET, CHANGE V.A.
052532  005303                        DEC     R3              ;MAKE R3 EQUAL TO PLF
052534  020103                        CMP     R1,R3           ;SEE IF PDR 4'S PLF=R3
052536  001414                        BEQ     6$              ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
052540  104051                        ERROR   51              ;ABORT WRONG PLACE
```

```
                                        BR      6$              ;BRANCH TO CHANGE PLF
                                4$:     CMP     R1,R3           ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
                                        BHIS    5$              ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
                                        ERROR   52              ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
                                        BR      6$              ;BRANCH TO CHANGE PLF
                                5$:     CMPB    R3,#177         ;SEE IF V.A. IS 177
                                        BEQ     6$              ;BRANCH TO CHECK PLF
                                        ADD     #100,R2         ;CHANGE VIRTUAL ADDRESS
                                        BR      2$              ;GO TRY THE NEXT VIRTUAL ADDRESS
                                6$:     CMPB    #177,(R0)       ;SEE IF PDR 4'S PLF IS 177
                                        BEQ     7$              ;BRANCH TO EXIT IF PLF IS 177
                                        INCB    (R0)            ;STEP PLF OF PDR 4 UP BY 1
                                        BR      1$              ;BRANCH TO START WITH V.A. OF 0
                                7$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

                                ;;*****************************************************************
                                ;*TEST 52       PAGE LENGTH FAULTS - DOWNWARD EXPANSION
                                ;*
                                ;*      THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS ON PAGE 'SAPL'
                                ;*      AND THE LOGIC THAT GENERATES 'SAPL LENGTH FAULT'.  IT TRIES
                                ;*      EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS.  THE
                                ;*      TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
                                ;*      OCCURS.  IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
                                ;*      THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
                                ;*
                                ;;*****************************************************************
052610                          TST52:
                                        SCOPE
052610  000004
052612  012737  053002  001314          MOV     #TST53,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
052620  012737  000024  001204          MOV     #24,$TIMES      ;;DO 24 ITERATIONS
052626  012737  077416  172310  20$:    MOV     #77416,@#KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
052634  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
052642  012700  172311                  MOV     #KIPDR4+1,R0    ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
052646  012737  052700  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
052654  005037  001246          1$:     CLR     PMMR0           ;CLEAR LOCATION THAT HOLDS MMR0
052660  012702  117700                  MOV     #117700,R2      ;PUT VIRTUAL ADDRESS INTO R2
052664  010203                  2$:     MOV     R2,R3           ;PUT VIRTUAL ADDRESS INTO R3 TOO
052666  072327  177772                  ASH     #-6,R3          ;RIGHT SHIFT 6 BITS SO IT WILL
                                                                ;MATCH THE PLF
052672  042703  177600                  BIC     #177600,R3      ;CLEAR BITS THAT ARE SET IN UPPER BYTE
052676  111001                          MOVB    (R0),R1         ;SAVE PLF IN R1 FOR LATER COMPARISON
052700  012737  040011  001224  3$:     MOV     #040011,MMEXP   ;POTENTIAL ABORT CONDITION: PAGE LENGTH
                                                                ;ERROR KERNEL, I-SPACE, PAGE 4
052706  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
052710  011204                          MOV     (R2),R4         ;READ USING V.A. IN R2
052712  005037  001224                  CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION
052716  005737  001246                  TST     PMMR0           ;SEE IF ABORT OCCURRED YET
052722  001405                          BEQ     4$              ;BRANCH IF NO ABORT YET, CHANGE V.A.
052724  005203                          INC     R3              ;MAKE R3 EQUAL TO PLF
052726  020103                          CMP     R1,R3           ;SEE IF PDR 4'S PLF=R3
052730  001414                          BEQ     6$              ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
052732  104051                          ERROR   51              ;ABORT WRONG PLACE
052734  000412                          BR      6$              ;BRANCH TO CHANGE PLF
052736  020103                  4$:     CMP     R1,R3           ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
052740  101402                          BLOS    5$              ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
```

```
                                        ERROR   52              ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
                                        BR      6$              ;BRANCH TO CHANGE PLF
                                5$:     CMPB    R3,#000         ;SEE IF V.A. IS 000
                                        BEQ     6$              ;BRANCH TO CHECK PLF
                                        SUB     #100,R2         ;CHANGE VIRTUAL ADDRESS
                                        BR      2$              ;GO TRY THE NEXT VIRTUAL ADDRESS
                                6$:     CMPB    #000,(R0)       ;SEE IF PDR 4'S PLF IS 000
                                        BEQ     7$              ;BRANCH TO EXIT IF PLF IS 000
                                        DECB    (R0)            ;STEP PLF OF PDR 4 UP BY 1
                                        BR      1$              ;BRANCH TO START WITH V.A. OF 0
                                7$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

                        ;;**************************************************************************
                        ;*TEST 53       ACCESS CONTROL FIELD = 0, 3, OR 7 (ABORT ALL ACCESSES)
                        ;*
                        ;*      THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
                        ;*      OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MMR0.
                        ;*      BITS <06:01> IN MMR0 WILL LOCK UP ALL AVAILABLE INFORMATION
                        ;*      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
                        ;*      IS KERNEL I-SPACE PAGE 5.  THE EXPECTED ERROR CODE IS 100013.
                        ;*
                        ;;**************************************************************************
053002                  TST53:
                                SCOPE
000004
053004  012737  053266  001314        MOV     #TST54,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
053012  012737  077406  172310  20$:   MOV     #77406,@#KIPDR4 ;LOAD ACF 6 INTO PDR 4
053020  012737  077400  172312        MOV     #77400,@#KIPDR5 ;LOAD ACF 0 INTO PDR5

053026  012737  001000  172350        MOV     #1000,@#KIPAR4  ;LOAD 16K INTO PAR4
053034  012737  001000  172352        MOV     #1000,@#KIPAR5  ;LOAD 16K INTO PAR5
053042  012700  100000                MOV     #100000,R0      ;LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
053046  012701  120000                MOV     #120000,R1      ;LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
053052  012702  161457                MOV     #161457,R2      ;LOAD DATA PATTERN INTO R2
053056  012737  053066  001110  11$:   MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
053064  005010                        CLR     (R0)            ;CLEAR MEMORY LOCATION 100000
053066  012737  100013  001224  1$:    MOV     #100013,MMEXP   ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
                                                                ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
053074  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
053076  010211                8$:     MOV     R2,(R1)         ;WRITE TO NON-RESIDENT OR UNUSED ACF
                                                                ;SHOULD CAUSE ABORT
053100  005037  001224                CLR     MMEXP           ;CLEAR EXPECTED ABORT CONDITION

053104  005710                        TST     (R0)            ;SEE IF (100000) IS STILL ZERO
053106  001401                        BEQ     2$              ;BRANCH IF (100000) IS ZERO
053110  104053                        ERROR   53              ;ABORT DID NOT HAPPEN
053112  012737  053124  001110  2$:    MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
053120  010210                        MOV     R2,(R0)         ;LOAD DATA PATTERN INTO 100000
053122  005004                        CLR     R4              ;CLEAR REGISTER TO RECEIVE DATA
053124  012737  100013  001224  4$:    MOV     #100013,MMEXP   ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
                                                                ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
053132  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
053134  011104                9$:     MOV     (R1),R4         ;TRY TO READ (100000) INTO R4
                                                                ;THIS SHOULD ABORT
053136  005037  001224                CLR     MMEXP           ;EXPECTED ABORT CONDITION
```

```
6658
6659  053142  005704                          TST     R4              ;MAKE SURE R4 IS STILL 0
6660  053144  001401                          BEQ     5$              ;BRANCH IF R4 IS 0
6661  053146  104053                          ERROR   53              ;ABORT DID NOT HAPPEN
6662  053150  023727  172312  077407  5$:     CMP     KIPDR5,#77407   ;SEE IF PDR 5'S ACF=7
6663  053156  001414                          BEQ     12$             ;TEST OVER IF ACF=7
6664  053160  023727  172312  077403          CMP     KIPDR5,#77403   ;SEE IF PDR 5'S ACF=3
6665  053166  001004                          BNE     10$             ;BRANCH TO MAKE ACF=3 IF NOT 3
6666  053170  012737  077407  172312          MOV     #77407,KIPDR5   ;MAKE ACF=7 IF ALREADY 3
6667  053176  000727                          BR      11$             ;REPEAT TEST WITH ACF=7
6668  053200  012737  077403  172312  10$:    MOV     #77403,KIPDR5   ;MAKE PDR 5'S ACF=3
6669  053206  000723                          BR      11$             ;REPEAT TEST WITH ACF=3
6670                                  ;;
6671                                  ;;      THIS SECTION OF CODE VERIFIES THAT YOU WON'T MODIFY A MEMORY
6672                                  ;;      MANAGEMENT REGISTER ON A M.M. ABORT REFERENCE.
6673                                  ;;
6674  053210  012737  053236  001110  12$:    MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
6675  053216  012737  177600  172352          MOV     #177600,KIPAR5  ;MAP PAGE 5 TO I/O PAGE
6676  053224  012701  132350                  MOV     #132350,R1      ;LOAD VIRTUAL ADDRESS TO REFERENCE
6677                                                                  ;KIPAR4
6678  053230  012737  100013  001224          MOV     #100013,MMEXP   ;EXPECTING NON-RESIDENT ABORT
6679                                                                  ;KERNEL I PAGE 5
6680  053236  000240                  13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6681  053240  005011                          CLR     (R1)            ;THIS INSTRUCTION SHOULD ABORT
6682                                                                  ;DURING THE ABORT 'SSRC INH T3' IS
6683                                                                  ;ASSERTED TO STOP THE CLOCKING OF THE
6684                                                                  ;FLIP/FLOPS ON SCCX
6685  053242  022737  001000  172350          CMP     #1000,KIPAR4    ;KIPAR4 SHOULD STILL BE 1000
6686  053250  001401                          BEQ     14$             ;BRANCH IF KIPAR4 IS STILL 1000
6687  053252  104127                          ERROR   127             ;ABORT DID NOT STOP CLRING OF KIPAR4
6688  053254  012737  053012  001110  14$:    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
6689  053262  005037  001224                  CLR     MMEXP           ;NOT EXPECTING ANY TRAPS
6690
6691
6692                                  ;;**************************************************************
6693                                  ;*TEST 54        ACCESS CONTROL FIELD = 2  (ABORT ON WRITE)
6694                                  ;*
6695                                  ;*      THIS IS A READ ONLY A.C.F., ANY WRITE ATTEMPT TO THIS PAGE
6696                                  ;*      WILL ABORT AND SET BIT 13 OF MMR0.
6697                                  ;*      BITS <06:01> IN MMR0 WILL LOCK UP ALL AVAILABLE INFORMATION
6698                                  ;*      ON THAT PAGE.  IN THIS CASE THE WRITE ATTEMPT WILL BE TO
6699                                  ;*      KERNEL I-SPACE PAGE 4.  THE EXPECTED ERROR CODE IS 020011.
6700                                  ;*
6701                                  ;;**************************************************************
6702  053266                          TST54:
6703  053266  000004                          SCOPE
6704  053270  012737  053416  001314          MOV     #TST55,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
6705                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
6706  053276                          20$:
6707  053276  012737  001000  172352          MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K
6708  053304  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
6709  053312  012737  077406  172312          MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
6710                                                                  ;EXPANDS UPWARD, AND IS READ/WRITE
6711                                                                  ;WITH NO TRAPPING
6712  053320  012737  077402  172310          MOV     #77402,KIPDR4   ;LOAD ACF 2 INTO PDR 4
6713  053326  005037  001224                  CLR     MMEXP           ;NOT EXPECTING ANY TRAPS OR ABORTS YET
```

```
6714  053332  012737  002222  120000          MOV     #2222,@#120000  ;LOAD DATA INTO 16K
6715  053340  013700  100000                  MOV     @#100000,R0     ;READ DATA THRU PAGE 4
6716  053344  012737  053356  001110  1$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
6717  053352  005037  120000                  CLR     @#120000        ;CLEAR TEST LOCATION THRU PAGE 5
6718  053356  012737  020011  001224  11$:    MOV     #20011,MMEXP    ;EXPECTING READ ONLY FAULT, PAGE 4
6719  053364  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6720  053366  012737  017777  100000          MOV     #17777,@#100000 ;TRY TO WRITE THRU PAGE 4
6721  053374  005037  001224                  CLR     MMEXP           ;NO MORE TRAPS EXPECTED
6722  053400  005737  120000                  TST     @#120000        ;SEE IF TEST LOCATION IS STILL ZERO
6723  053404  001401                          BEQ     2$              ;BRANCH IF WORD IS STILL ZERO
6724  053406  104054                          ERROR   54              ;NO ABORT ON PAGE 4
6725  053410  012737  053276  001110  2$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
6726
6727
6728                                  ;;*****************************************************************
6729                                  ;*TEST 55       ACCESS CONTROL FIELD = 1  (ABORT ON WRITE, TRAP ON READ)
6730                                  ;*
6731                                  ;*      THIS IS ANOTHER READ ONLY A.C.F., ALL WRITES TO THIS PAGE WILL
6732                                  ;*      ABORT AND SET BIT 13 OF MMR0.
6733                                  ;*      BITS <06:01> IN MMR0 WILL LOCK UP ALL AVAILABLE INFORMATION
6734                                  ;*      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
6735                                  ;*      IS KERNEL I-SPACE PAGE 4.  THE EXPECTED ERROR CODE IS 020011.
6736                                  ;*
6737                                  ;*      IF BIT 09 OF MMR0 (ENABLE MEMORY MANAGEMENT TRAPS) IS SET
6738                                  ;*      THEN ALL READS TO THIS PAGE WILL TRAP, AFTER THE INSTRUCTION
6739                                  ;*      IS COMPLETED, SETTING BIT 12 OF MMR0.
6740                                  ;*
6741                                  ;*      AFTER THE ABORT ON WRITE IS TESTED, A READ FROM THIS PAGE
6742                                  ;*      WITH BIT 9 CLEAR WILL BE DONE TO ENSURE THAT TRAPPING DOESN'T
6743                                  ;*      TAKE PLACE WHEN NOT ENABLED.  THEN BIT 09 IS SET AND ANOTHER
6744                                  ;*      READ IS DONE (THIS TIME IT SHOULD TRAP TO PAGE 1 KERNEL MODE).
6745                                  ;*
6746                                  ;*****************************************************************
6747  053416                          TST55:
6748  053416  000004                          SCOPE
6749  053420  012737  053662  001314          MOV     #TST56,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
6750                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
6751  053426                          20$:
6752  053426  012737  001000  172352          MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K
6753  053434  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
6754  053442  012737  077406  172312          MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
6755                                                                  ;EXPANDS UPWARD, AND IS READ/WRITE
6756                                                                  ;WITH NO TRAPPING
6757  053450  012737  077401  172310          MOV     #77401,KIPDR4   ;SET ACF = 1 FOR PAGE 4
6758  053456  012737  053470  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
6759  053464  005037  001246                  CLR     PMMR0           ;CLEAR M.M. ABORT FLAG
6760  053470  012737  020011  001224  10$:     MOV     #20011,MMEXP    ;READ ONLY ABORT, PAGE 4
6761  053476  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6762  053500  012737  017777  100000          MOV     #17777,@#100000 ;TRY TO WRITE THRU PAGE 4
6763  053506  005037  001224                  CLR     MMEXP           ;NO MORE TRAPS EXPECTED
6764  053512  005737  001246                  TST     PMMR0           ;SEE IF M.M. ABORT HAPPENED
6765  053516  001001                          BNE     1$              ;BRANCH IF ABORT HAPPENED
6766  053520  104054                          ERROR   54              ;NO ABORT ON PAGE 4 A.C.F.=1
6767  053522  012737  053540  001110  1$:     MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
6768  053530  012700  012547                  MOV     #12547,R0       ;PUT DATA PATTERN INTO R0
6769  053534  010037  120000                  MOV     R0,@#120000     ;LOAD DATA PATTERN THRU PAGE 5
```

```
 6770  053540  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 6771  053542  013701  100000                  MOV     @#100000,R1     ;READ TEST LOCATION THRU PAGE 4
 6772  053546  020100                          CMP     R1,R0           ;SEE IF DATA READ TOOK PLACE
 6773  053550  001401                          BEQ     2$              ;BRANCH IF READ CORRECT
 6774  053552  104055                          ERROR   55              ;INCORRECT READ, BIT09 (MMR0) IS CLEAR
 6775  053554  013737  177572  001246  2$:     MOV     MMR0,PMMR0      ;SAVE MMR0 IN CASE OF ERROR
 6776  053562  032737  010000  001246          BIT     #BIT12,PMMR0    ;BIT 12 SHOULD SET EVEN IF YOU DON'T
 6777                                                                  ;TAKE THE TRAP DUE TO BIT09 BEING CLR
 6778  053570  001001                          BNE     3$              ;BRANCH IF BIT12 IS SET
 6779  053572  104066                          ERROR   66              ;BIT 12 WAS NOT SET
 6780  053574  012737  053604  001110  3$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
 6781  053602  005001                          CLR     R1              ;CLEAR REG 1, RECEIVES DATA ON FETCH
 6782  053604  005037  001246          12$:    CLR     PMMR0           ;CLEAR M.M. TRAP FLAG
 6783  053610  012737  001001  177572          MOV     #1001,MMR0      ;ENABLE TRAPS, FULL RELOCATION, CLR BIT 12
 6784  053616  012737  011003  001224          MOV     #11003,MMEXP    ;EXPECTED TRAP CONDITION
 6785                                                                  ;TRAP, TRAPS ENABLED, RELOCATION
 6786  053624  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 6787  053626  013701  100000                  MOV     @#100000,R1     ;READ THRU PAGE 4
 6788  053632  005037  001224                  CLR     MMEXP           ;NO MORE TRAPS EXPECTED
 6789  053636  005737  001246                  TST     PMMR0           ;SEE IF TRAP OCCURED
 6790  053642  001001                          BNE     4$              ;BRANCH IF TRAP OCCURRED
 6791  053644  104056                          ERROR   56              ;NO TRAP
 6792  053646  020001                  4$:     CMP     R0,R1           ;SEE IF DATA READ WAS CORRECT
 6793  053650  001401                          BEQ     5$              ;BRANCH IF READ CORRECT
 6794  053652  104057                          ERROR   57              ;INCORRECT READ, BIT09 (MMR0) WAS SET
 6795  053654  012737  053426  001110  5$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 6796
 6797
```

```
                                ;;*****************************************************************
                                ;*TEST 56       ACCESS CONTROL FIELD = 4  (TRAP ON READ OR WRITE)
                                ;*
                                ;*      THIS A.C.F. IS READ/WRITE BUT ALL REFERENCES TO THIS PAGE
                                ;*      WILL TRAP TO VECTOR 250 SETTING BIT 12 OF MMR0 IF BIT 9
                                ;*      (ENABLE MEMORY MANAGEMENT TRAPS) IS SET.
                                ;*
                                ;*      SINCE I HAVE ALREADY TESTED THE FACT THAT BIT 9 OF MMR0 DOES
                                ;*      INDEED ENABLE M.M. TRAPS I WILL JUST SET BIT 9 AND VERIFY THAT
                                ;*      BOTH A READ AND A WRITE TO PAGE 4 A.C.F. = 4 TRAP CORRECTLY
                                ;*
                                ;;*****************************************************************
053662                          TST56:
053662  000004                          SCOPE
053664  012737  054102  001314          MOV     #TST57,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
053672                          20$:
053672  012737  001000  172352          MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K
053700  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
053706  012737  077406  172312          MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
                                                                ;EXPANDS UPWARD, AND IS READ/WRITE
                                                                ;WITH NO TRAPPING
053714  012737  077404  172310          MOV     #77404,KIPDR4   ;SET ACF = 4 IN PDR 4
053722  012700  013451                  MOV     #13451,R0       ;LOAD DATA PATTERN INTO R0
053726  010037  120000                  MOV     R0,@#120000     ;LOAD DATA INTO TEST LOCATION
053732  005001                          CLR     R1              ;WHAT DO YOU THINK
053734  012737  053742  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
053742  005037  001246          10$:    CLR     PMMR0           ;CLEAR FLAG LOCATION
053746  012737  011003  001224          MOV     #11003,MMEXP    ;EXPECTING TRAP WITH TRAPS ENABLED
                                                                ;KERNEL I PAGE 1, FULL RELOCATION
053754  012737  001001  177572          MOV     #1001,MMR0      ;ENABLE M. M. TRAPS
053762  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
053764  013701  100000                  MOV     @#100000,R1     ;TRY TO READ THRU PAGE 4
053770  005037  001224                  CLR     MMEXP           ;NO MORE TRAPS EXPECTED
053774  005737  001246                  TST     PMMR0           ;SEE IF TRAP OCCURRED
054000  001001                          BNE     1$              ;BRANCH IF TRAP
054002  104056                          ERROR   56              ;NO TRAP
054004  020001                  1$:     CMP     R0,R1           ;SEE IF READ WAS CORRECT
054006  001401                          BEQ     2$              ;BRANCH IF CORRECT
054010  104057                          ERROR   57              ;INCORRECT READ, BIT09 (MMR0) WAS SET
054012  012737  054020  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
054020  005037  001246          12$:    CLR     PMMR0           ;CLEAR FLAG LOCATION
054024  012737  001001  177572          MOV     #1001,MMR0      ;ENABLE TRAPPING
054032  012737  011003  001224          MOV     #11003,MMEXP    ;EXPECTING TRAP WITH TRAPS ENABLED
                                                                ;KERNEL I PAGE 1, FULL RELOCATON
054040  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
054042  012737  000000  100000          MOV     #0,@#100000     ;TRY TO WRITE INTO PAGE 4
054050  005037  001224                  CLR     MMEXP           ;NOT EXPECTING ANY TRAPS
054054  005737  001246                  TST     PMMR0           ;SEE IF TRAP OCCURRED
054060  001001                          BNE     3$              ;BRANCH IF TRAP
054062  104056                          ERROR   56              ;NO TRAP
054064  005737  120000          3$:     TST     @#120000        ;SEE IF WRITE OCCURED
054070  001401                          BEQ     4$              ;BRANCH IF WRITE HAPPENED
054072  104060                          ERROR   60              ;NO WRITE, BIT09 (MMR0) WAS SET
054074  012737  053672  001110  4$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                        ;;*********************************************************************
                        ;*TEST 57        ACCESS CONTROL FIELD = 5  (TRAP ON WRITE)
                        ;*
                        ;*      THIS TEST IS RUN WITH THE ENABLE M.M. TRAPS BIT (BIT09) SET.
                        ;*      THE A.C.F. FOR PAGE 4 IS SET TO 5 (TRAP ON WRITE).
                        ;*      A READ FROM PAGE 4 IS TRIED EXPECTING NO TRAP AND THEN A WRITE
                        ;*      TO PAGE 4 IS TRIED EXPECTING A M.M. TRAP.  THE TRAP IS VERIFIED
                        ;*      BY THE USE OF A TRAP FLAG WHICH IS SET IN THE TRAP ROUTINE.
                        ;*
                        ;;*********************************************************************
054102                  TST57:
                                SCOPE
054102  000004
054104  012737  054300  001314        MOV     #TST60,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
054112                  20$:
054112  012737  001000  172352        MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K
054120  012737  001000  172350        MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
054126  012737  077406  172312        MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
                                                                ;EXPANDS UPWARD, AND IS READ/WRITE
                                                                ;WITH NO TRAPPING
054134  012737  077405  172310        MOV     #77405,KIPDR4   ;SET ACF = 5 IN PDR 4
054142  012700  012345                MOV     #12345,R0       ;SET DATA PATTERN INTO R0
054146  010037  120000                MOV     R0,@#120000     ;LOAD TEST LOCATION WITH DATA
054152  005037  001224                CLR     MMEXP           ;NO TRAP EXPECTED
054156  005001                        CLR     R1              ;CLEAR REGISTER 1
054160  012737  054166  001110        MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
054166  012737  001001  177572  10$:  MOV     #1001,MMR0      ;ENABLE M M TRAPS
054174  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
054176  013701  100000                MOV     @#100000,R1     ;READ TEST LOCATION THRU PAGE 4
054202  020001                        CMP     R0,R1           ;SEE IF READ WAS CORRECT
054204  001401                        BEQ     1$              ;BRANCH IF READ CORRECT
054206  104057                        ERROR   57              ;INCORRECT READ, BIT09 (MMR0) WAS SET
054210  012737  054216  001110  1$:   MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
054216  005037  001246        11$:    CLR     PMMR0           ;CLEAR FLAG LOCATION
054222  012737  001001  177572        MOV     #1001,MMR0      ;ENABLE TRAPS
054230  012737  011003  001224        MOV     #11003,MMEXP    ;EXPECTING M.M. TRAP

054236  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
054240  012737  000000  100000        MOV     #0,@#100000     ;TRY TO WRITE INTO PAGE 4
054246  005037  001224                CLR     MMEXP           ;NO MORE TRAPS EXPECTED
054252  005737  001246                TST     PMMR0           ;SEE IF TRAP OCCURRED
054256  001001                        BNE     2$              ;BRANCH IF TRAP
054260  104056                        ERROR   56              ;NO TRAP
054262  005737  120000        2$:     TST     @#120000        ;SEE IF WRITE OCCURRED
054266  001401                        BEQ     3$              ;BRANCH IF WRITE HAPPENED
054270  104060                        ERROR   60              ;NO WRITE, BIT09 (MMR0) WAS SET
054272  012737  054112  001110  3$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                        ;;*********************************************************************
                        ;*TEST 60        NO TRAP WHEN TRAP BIT IS SET
                        ;*
                        ;*      THIS TEST VERIFIES THE LOGIC (ON 'SSRD') THAT PREVENTS A M.M.
                        ;*      TRAP AS LONG AS BIT12 OF MMR0 (M.M. TRAP BIT) IS SET.  THE
                        ;*      TEST SETS BITS 12, 09, & 00 OF MMR0 AND TRIES A REFERENCE TO
```

```
                                ;*      PAGE 4 WHOSE A.C.F.= 4 (TRAP ON ALL REFERENCES).  NO TRAP IS
                                ;*      EXPECTED, SO IF THE LOGIC FAILS AN UNEXPECTED M.M. TRAP WILL
                                ;*      OCCUR.
                                ;;*****************************************************************
054300                          TST60:
054300  000004                          SCOPE
054302  012737  054422  001314        MOV     #TST61,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
054310                          20$:
054310  012737  001000  172352        MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K
054316  012737  001000  172350        MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
054324  012737  077406  172312        MOV     #77406,KIPDR5   ;PAGE 5 IS 200 BLOCKS LONG,
                                                                ;EXPANDS UPWARD, AND IS READ/WRITE
                                                                ;WITH NO TRAPPING
054332  012737  077404  172310        MOV     #77404,KIPDR4   ;SET ACF = 4 IN PDR 4
054340  005037  001224                CLR     MMEXP           ;NO TRAPS EXPECTED
054344  012737  017777  120000        MOV     #17777,@#120000 ;LOAD DATA INTO TEST LOCATION
054352  012737  054360  001110        MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
054360  012737  011001  177572  10$:   MOV     #11001,MMR0     ;ENABLE TRAPS AND SET TRAP BIT (12)
054366  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
054370  012737  000000  100000        MOV     #0,@#100000     ;TRY TO WRITE THRU PAGE 4
054376  012737  000001  177572        MOV     #1,MMR0         ;CLEAR BITS 9 & 12 OF MMR0
054404  005737  120000                TST     @#120000        ;SEE IF WORD WAS CHANGED
054410  001401                        BEQ     1$              ;BRANCH IF LOCATION IS CLEAR
054412  104060                        ERROR   60              ;NO WRITE, BIT09 (MMR0) WAS SET
054414  012737  054310  001110  1$:   MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST


                                ;;*****************************************************************
                                ;*TEST 61        NO TRAPPING WHEN REFERENCING A MEMORY MANAGEMENT REG
                                ;*
                                ;*      THIS VERIFIES THE LOGIC THAT PREVENTS A M.M. TRAP WHEN
                                ;*      REFERENCING A M.M. REGISTER.  PAGE 7 IS MAPPED TO THE I/O
                                ;*      PAGE AND ITS A.C.F.= 4 (TRAP ON ALL REFERENCES).  EACH STATUS
                                ;*      REGISTER AND EACH PAR7 AND PDR7 IS READ THRU PAGE 7.  IF ANY
                                ;*      TRAPS OCCUR AN UNEXPECTED M.M. TRAP IS REPORTED.  THEN A MAP
                                ;*      REGISTER (170200) IS REFERENCED AND THE CORRECT TRAP IS VERIFIED
                                ;*      TO INSURE THAT A TRAP CAN OCCUR ON PAGE 7.
                                ;*      THE SIGNAL UNDER TEST IS 'SCCC INT REG B L'.
                                ;*
                                ;;*****************************************************************
054422                          TST61:
054422  000004                          SCOPE
054424  012737  054652  001314        MOV     #TST62,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
054432                          20$:
054432  012700  012754                MOV     #12754,R0       ;LOAD DATA PATTERN INTO R0
054436  010037  170200                MOV     R0,MAPL0        ;LOAD MAP REGISTER 0
054442  005037  001224                CLR     MMEXP           ;NOT EXPECTING ANY TRAPS
054446  012737  077404  172316        MOV     #77404,KIPDR7   ;SET ACF = 4 IN PAGE 7
054454  012737  001001  177572        MOV     #1001,MMR0      ;ENABLE MEMORY MANAGEMENT TRAPS
054462  013701  172356                MOV     KIPAR7,R1       ;READ KERNEL PAR 7
054466  013701  172256                MOV     SIPAR7,R1       ;READ SUPERVISOR PAR 7
054472  013701  177656                MOV     UIPAR7,R1       ;READ USER PAR 7
054476  013701  172316                MOV     KIPDR7,R1       ;READ KERNEL PDR 7
054502  013701  172216                MOV     SIPDR7,R1       ;READ SUPERVISOR PDR 7
```

```
6966  054506  013701  177616              MOV     UIPDR7,R1       ;READ USER PDR 7
6967  054512  013701  177572              MOV     MMR0,R1         ;READ MMR0
6968  054516  013701  177574              MOV     MMR1,R1         ;READ MMR1
6969  054522  013701  177576              MOV     MMR2,R1         ;READ MMR2
6970  054526  013701  172516              MOV     MMR3,R1         ;READ MMR3
6971  054532  012737  001001  177572      MOV     #1001,MMR0      ;MAKE SURE TRAPS ARE ENABLED
6972  054540  005037  001246              CLR     PMMR0           ;CLEAR M.M. TRAP FLAG
6973  054544  012737  011003  001224      MOV     #11003,MMEXP    ;EXPECTING M.M. TRAP ON NEXT REFERENCE
6974  054552  013701  170200              MOV     MAPL0,R1        ;TRY TO READ MAP REGISTER 0
6975  054556  005037  001224              CLR     MMEXP           ;NOT EXPECTING ANY M.M. TRAPS
6976  054562  005737  001246              TST     PMMR0           ;SEE IF TRAP OCCURRED
6977  054566  001001                      BNE     1$              ;BRANCH IF TRAP OCCURED
6978  054570  104061                      ERROR   61              ;NO TRAP
6979  054572  020001              1$:     CMP     R0,R1           ;SEE IF DATA WAS READ CORRECTLY
6980  054574  001401                      BEQ     2$              ;BRANCH IF DATA IS RIGHT
6981  054576  104062                      ERROR   62              ;INCORRECT READ ON I/O PAGE
6982  054600  012737  001000  177572  2$: MOV     #BIT9,MMR0      ;ENABLE TRAPS, NO RELOCATION
6983  054606  005037  001246              CLR     PMMR0           ;CLEAR M.M. TRAPS FLAG
6984  054612  000240                      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6985  054614  013701  170200              MOV     MAPL0,R1        ;READ MAP REGISTER 0
6986                                                              ;THIS READ SHOULD NOT TRAP SINCE
6987                                                              ;THERE IS NO RELOCATION ENABLED.
6988  054620  005737  001246              TST     PMMR0           ;SEE IF TRAP OCCURRED
6989  054624  001401                      BEQ     3$              ;BRANCH IF NO TRAP
6990  054626  104063                      ERROR   63              ;TRAPPED WHEN NO RELOCATION ENABLED
6991  054630  012737  000001  177572  3$: MOV     #BIT0,MMR0      ;ENABLE FULL RELOCATION
6992  054636  012737  077406  172316      MOV     #77406,KIPDR7   ;SET PDR 7 TO NO TRAPPING ACF
6993  054644  012737  054432  001110      MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
6994
6995
6996                              ;;*****************************************************************
6997                              ;*TEST 62       ONLY ONE VECTOR TAKEN IF TRAP AND ABORT
6998                              ;*
6999                              ;*      IF THERE IS A M.M. TRAP CONDITION AND A M.M. ABORT ON THE
7000                              ;*      SAME INSTRUCTION ONLY ONE VECTOR TO 000250 SHOULD BE TAKEN.
7001                              ;*      'SSRC KT ABORT FLG L' AND 'SSRC ABT FLG (0) H' WILL KNOCK
7002                              ;*      DOWN 'SSRD MEM MGMT TRAP L' SO THAT ONLY THE ABORT VECTOR WILL
7003                              ;*      BE TAKEN.
7004                              ;*      THIS TEST SETS THE VECTOR TO THE CODE AT 10$ AND, IF TWO VECTORS
7005                              ;*      ARE TAKEN ERROR 64 IS CALLED.  MMR0 SHOULD REPORT BOTH THE TRAP
7006                              ;*      AND THE ABORT CONDITIONS WHICH ARE: PAGE LENGTH, KERNEL I-SPACE
7007                              ;*      PAGE 5, AND BIT12 OF MMR0.  (051013)
7008                              ;*
7009                              ;;*****************************************************************
7010  054652                      TST62:
7011  054652  000004                     SCOPE
7012  054654  012737  055050  001314      MOV     #TST63,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
7013                                                              ;TEST FOR ESCAPE ON PARITY ERRORS
7014  054662  012737  054726  001110 20$: MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
7015  054670  012737  054750  000250      MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
7016  054676  012737  001000  172350      MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K - 20K
7017  054704  012737  001000  172352      MOV     #1000,KIPAR5    ;MAP PAGE 5 TO 16K- 20K
7018  054712  012737  000006  172312      MOV     #000006,KIPDR5  ;SET PAGE LENGTH TO ONE FOR PAGE 5
7019  054720  012737  077404  172310      MOV     #77404,KIPDR4   ;SET TRAP ON READ OR WRITE FOR PAGE 4
7020  054726  012706  001100          1$: MOV     #KERSTK,KSP     ;MAKE SURE KERN STK PTR IS SETUP IN
7021                                                              ;CASE YOU LOOP ON ERROR
```

```
7022  054732  012737  001001  177572          MOV     #1001,MMR0          ;ENABLE M.M. TRAPS
7023  054740  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
7024  054742  013737  100000  120100          MOV     @#100000,@#120100   ;TRY TO READ THRU PAGE 4 AND
7025                                                                      ;WRITE THRU PAGE 5 (PAGE 4 TRAP &
7026                                                                      ;PAGE 5 ABORT PAGE LENGTH)
7027  054750  020627  001074          10$:    CMP     KSP,#1074           ;HAS THE KERNEL STACK ONLY BEEN
7028                                                                      ;PUSHED ONCE BY THE PREVIOUS INSTRUCTION
7029  054754  001404                          BEQ     12$                 ;BRANCH IF IT HAS BEEN PUSHED
7030                                                                      ;ONLY ONE TIME
7031  054756  010637  001170                  MOV     KSP,$TMP0           ;SAVE THE KERNEL STACK POINTER FOR TYPE OUT
7032  054762  104064                          ERROR   64                  ;TWO PUSHES WHEN ONLY ONE SHOULD HAPPEN
7033  054764  000413                          BR      15$                 ;BRANCH TO EXIT TEST
7034  054766  013737  177572  001246  12$:    MOV     MMR0,PMMR0          ;SAVE MMR0 FOR CHECK
7035  054774  012737  051013  001224          MOV     #51013,MMEXP        ;MMR0 SHOULD HAVE PAGE LENGTH,
7036                                                                      ;TRAP, ENABLE TRAP, PAGE 5, RELOCATING
7037  055002  023737  001246  001224          CMP     PMMR0,MMEXP         ;SEE IF ABORT CONDITION IS CORRECT
7038  055010  001401                          BEQ     15$                 ;BRANCH TO EXIT IF CORRECT
7039  055012  104065                          ERROR   65                  ;INCORRECT ABORT CONDITION
7040  055014  012716  055022          15$:    MOV     #16$,(KSP)          ;CHANGE RETURN ADDRESS TO 16$
7041  055020  000006                          RTT                         ;RETURN TO 16$ AND CONTINUE PROGRAM
7042  055022  012737  025054  000250  16$:    MOV     #MMTRAP,MMVEC       ;RESTORE TRAP HANDLER
7043  055030  012737  000001  177572          MOV     #BIT0,MMR0          ;CLEAR OUT MMR0, BUT LEAVE RELOC ON
7044  055036  005037  001224                  CLR     MMEXP               ;NOT EXPECTING ANY M.M. TRAPS
7045  055042  012737  054662  001110          MOV     #20$,$LPERR         ;SET LOOP POINTER TO START OF TEST
7046
7047
7048                                  ;;*****************************************************************
7049                                  ;*TEST 63       PROPER TIMING OF MEMORY MANAGEMENT TRAPS
7050                                  ;*
7051                                  ;*      IF THE INSTRUCTION SETTING BIT09 OF MMR0 SATISFIES A M.M. TRAP
7052                                  ;*      CONDITION, NO TRAP SHOULD OCCUR SINCE BIT09 WILL NOT BE SET
7053                                  ;*      WHEN THE TRAP CONDITION IS SATISFIED.
7054                                  ;*      THE SECOND HALF OF THIS TEST VERIFIES THAT IF THE M.M. TRAP
7055                                  ;*      CONDITION IS MET DURING THE INSTRUCTION WHICH CLEARS BIT 09
7056                                  ;*      OF MMR0 THE TRAP WILL OCCUR ANYWAY.
7057                                  ;*
7058                                  ;;*****************************************************************
7059  055050                          TST63:
7060  055050  000004                          SCOPE
7061  055052  012737  055222  001314          MOV     #TST64,NXTTST       ;SAVE STARTING ADDRESS OF NEXT
7062                                                                      ;TEST FOR ESCAPE ON PARITY ERRORS
7063  055060  012737  000000  172350  20$:    MOV     #000,KIPAR4         ;MAP PAGE 4 TO 0 - 4K
7064  055066  012737  077404  172310          MOV     #77404,KIPDR4       ;TRAP ALL REFERENCES
7065  055074  012737  055116  001110          MOV     #1$,$LPERR          ;SET LOOP ON ERROR POINTER TO 1$
7066  055102  012700  001354                  MOV     #ENMMTR,R0          ;PUT ADDRESS OF ENABLE M.M. TRAPS
7067                                                                      ;WORD INTO R0
7068  055106  052700  100000                  BIS     #BIT15,R0           ;MAKE ADDRESS IN R0 USE PAGE 4
7069  055112  005037  001224                  CLR     MMEXP               ;NOT EXPECTING ANY TRAPS ON THIS REF.
7070  055116  000240                  1$:     NOP                         ;THIS IS A SYNC POINT FOR SCOPING
7071  055120  051037  177572          11$:    BIS     (R0),MMR0           ;SET ENABLE TRAPS BIT IN MMR0 USING
7072                                                                      ;PAGE THAT COULD CAUSE TRAP IF BIT09
7073                                                                      ;WERE ON DURING SOURCE MODE
7074  055124  013737  177572  001246          MOV     MMR0,PMMR0          ;READ MMR0 FOR CHECK ON BIT 12
7075  055132  032737  010000  001246          BIT     #BIT12,PMMR0        ;SEE IF BIT12 IS SET BY INST. AT 11$
7076  055140  001001                          BNE     2$                  ;BRANCH IF IT IS SET
7077  055142  104066                          ERROR   66                  ;BIT 12 NOT SET IN MMR0
```

```
 7078  055144  012737  055160  001110  2$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
 7079  055152  012737  010003  001224          MOV     #10003,MMEXP    ;EXPECTING TRAP, BUT ENABLE TRAPS BIT
 7080                                                                  ;SHOULD BE CLEAR BEFORE END OF INST.
 7081                                                                  ;THAT CAUSES THE TRAP
 7082  055160  012737  001001  177572  3$:     MOV     #1001,MMR0      ;ENABLE M.M. TRAPS, FULL RELOCATION
 7083  055166  005037  001246                  CLR     PMMR0           ;CLEAR M.M. TRAP FLAG
 7084  055172  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 7085  055174  041037  177572                  BIC     (R0),MMR0       ;CLEAR ENABLE M.M. TRAP BIT AT END
 7086                                                                  ;OF INSTRUCTION, TRAP FLIP/FLOP SHOULD
 7087                                                                  ;BE SET DURING SOURCE MODE FETCH
 7088  055200  005737  001246                  TST     PMMR0           ;SEE IF TRAP REALLY OCCURRED ON LAST INS
 7089  055204  001001                          BNE     4$              ;BRANCH IF TRAP OCCURRED
 7090  055206  104067                          ERROR   67              ;NO TRAP, WHEN CLEARING BIT09 (MMR0)
 7091  055210  005037  001224          4$:     CLR     MMEXP           ;NO M.M. TRAPS EXPECTED
 7092  055214  012737  055060  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 7093
 7094
 7095
 7096                                  ;;*****************************************************************
 7097                                  ;*TEST 64        ABORT ON ILLEGAL MODE
 7098                                  ;*
 7099                                  ;*      IF THE MODE SET IN BITS <15:14> OF THE PROCESSOR STATUS IS
 7100                                  ;*      <10> THE MODE IS ILLEGAL AND THE NEXT INSTRUCTION FETCH WILL
 7101                                  ;*      SELECT NO PAR/PDR PAIR TO CONTROL THE REFERENCE.  THE PDR
 7102                                  ;*      LINES WILL ALL BE READ AS ONES.  THE A.C.F. = 7 (NON-RESIDENT),
 7103                                  ;*      THE EXPANSION DIRECTION = DOWN, THE P.L.F. = 177 OR 1 BLOCK.
 7104                                  ;*      THE M.M. ABORT WILL BE NON-RESIDENT, PAGE LENGTH (IF THE
 7105                                  ;*      VIRTUAL ADDRESS HAS A BLOCK NUMBER OF 176 OR LESS), MODE <10>,
 7106                                  ;*      PAGE 2 (SINCE THE CODE IS ON PAGE 2).
 7107                                  ;*
 7108                                  ;;*****************************************************************
 7109  055222                          TST64:
 7110  055222  000004                          SCOPE
 7111  055224  012737  055334  001314          MOV     #TST65,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
 7112                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
 7113  055232  012737  055260  000250  20$:     MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
 7114  055240  012737  055246  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
 7115  055246  000240                  1$:     NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 7116  055250  052737  100000  177776          BIS     #BIT15,PSW      ;SET ILLEGAL MODE IN PROCESSOR STATUS
 7117  055256  104071                          ERROR   71              ;INSTRUCTION FETCH DIDN'T ABORT
 7118                                                                  ;THIS INSTRUCTION FETCH SHOULD ABORT,
 7119                                                                  ;NON-RESIDENT & PAGE LENGTH FAULT,
 7120                                                                  ;ILLEGAL MODE, PAGE 1.
 7121
 7122
 7123  055260  013737  177572  001246  10$:     MOV     MMR0,PMMR0      ;READ MMR0 FOR COMPARE
 7124  055266  012716  055302                  MOV     #16$,(KSP)      ;CHANGE RETURN ADDRESS TO 16$
 7125  055272  042766  100000  000002          BIC     #BIT15,2(KSP)   ;CLEAR ILLEGAL MODE BIT IN PSW ON STACK
 7126  055300  000006                          RTT                     ;RETURN TO 16$ AND CONTINUE PROGRAM
 7127  055302  012701  140105          16$:     MOV     #140105,R1      ;LOAD EXPECTED ABORT CONDITION IN R1:
 7128                                                                  ;NON-RESIDENT, PAGE FAULT, MODE=<10>,
 7129                                                                  ;PAGE 2.
 7130  055306  020137  001246                  CMP     R1,PMMR0        ;DID YOU GET THE EXPECTED CONDITION
 7131  055312  001401                          BEQ     11$             ;BRANCH IF CONDITION IS CORRECT
 7132  055314  104072                          ERROR   72              ;WRONG ERROR CONDITION
 7133  055316  000237                  11$:     SPL     7               ;MAKE THE PRIORITY LEVEL 7
```

```
7134  055320  042737  177776  177572          BIC     #177776,MMR0    ;CLEAR ALL ERROR CONDITIONS
7135  055326  012737  055232  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
7136
7137
7138                                  ;;*****************************************************************
7139                                  ;*TEST 65       MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMR0 NOT CLEARED
7140                                  ;*
7141                                  ;*      AS LONG AS 'SSRC NO ERROR (1) H' IS NOT ASSERTED (THAT IS AFTER
7142                                  ;*      AN ABORT AND UNTIL MMR0 BITS <15:13> ARE CLEARED) MMR0, MMR1,
7143                                  ;*      AND MMR2 SHOULD NOT BE CLOCKED.  THIS TEST CAUSES A NON-RESIDENT
7144                                  ;*      ABORT, SAVES THE STATUS REGISTERS, CHANGES THE VECTOR TO 10$,
7145                                  ;*      AND THEN CAUSES A PAGE LENGTH ABORT.  AT THE SECOND ABORT THE
7146                                  ;*      STATUS REGISTERS ARE COMPARED WITH THEIR FIRST CONDITIONS. IF ANY
7147                                  ;*      OF THEM CHANGE, THE OLD AND THE NEW CONDITIONS WILL BE REPORTED.
7148                                  ;*
7149                                  ;;*****************************************************************
7150  055334                          TST65:
7151  055334  000004                          SCOPE
7152  055336  012737  055576  001314          MOV     #TST66,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
7153                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
7154  055344  012737  000000  172310  20$:    MOV     #000000,KIPDR4  ;MAP PAGE 4 NON-RESIDENT, AND PAGE
7155                                                                  ;LENGTH OF 1 BLOCK
7156  055352  012737  055414  000250          MOV     #5$,MMVEC       ;SET M.M. TRAP VECTOR TO 5$
7157  055360  012737  000340  000252          MOV     #340,MMVEC+2    ;SET PRIORITY TO 7 GOTO KERNEL MODE
7158  055366  012737  055374  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
7159  055374  013700  100000          1$:     MOV     @#100000,R0     ;TRY TO READ THRU PAGE 4
7160                                                                  ;THIS PAGE NON-RESIDENT SHOULD CAUSE
7161                                                                  ;ABORT AND TRAP TO 5$.
7162  055400  012737  055444  000250  2$:     MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
7163  055406  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7164  055410  013700  100100                  MOV     @#100100,R0     ;TRY TO READ FROM BLOCK 2 OF PAGE 4
7165                                                                  ;THIS SHOULD ABORT AGAIN BUT NONE
7166                                                                  ;OF THE MEMORY MANAGEMENT STATUS
7167                                                                  ;REGISTERS SHOULD BE CLOCKED THIS TIME.
7168
7169
7170  055414  013737  177572  001246  5$:     MOV     MMR0,PMMR0      ;READ MEMORY MANAGEMENT REGISTER 0
7171  055422  013737  177574  001250          MOV     MMR1,PMMR1      ;READ MEMORY MANAGEMENT REGISTER 1
7172  055430  013737  177576  001252          MOV     MMR2,PMMR2      ;READ MEMORY MANAGEMENT REGISTER 2
7173  055436  012716  055400                  MOV     #2$,(KSP)       ;CHANGE RETURN ADDRESS TO 2$
7174  055442  000006                          RTT                     ;GO BACK TO 2$ AND CAUSE PAGE FAULT
7175
7176
7177  055444  012716  055452          10$:    MOV     #16$,(KSP)      ;CHANGE RETURN ADDRESS TO 16$
7178  055450  000006                          RTT                     ;RETURN TO 16$ AND CONTINUE PROGRAM
7179  055452  005037  001176          16$:    CLR     $TMP3           ;ERROR COUNTER, 3 POSSIBLE CMP FAILURES
7180  055456  013737  177572  001170          MOV     MMR0,$TMP0      ;READ MEMORY MANAGEMENT REGISTER 0
7181  055464  013737  177574  001172          MOV     MMR1,$TMP1      ;READ MEMORY MANAGEMENT REGISTER 1
7182  055472  013737  177576  001174          MOV     MMR2,$TMP2      ;READ MEMORY MANAGEMENT REGISTER 2
7183  055500  023737  001174  001252          CMP     $TMP2,PMMR2     ;SEE IF MMR2 CHANGED
7184  055506  001402                          BEQ     11$             ;BRANCH IF MMR2 DIDN'T CHANGE
7185  055510  005237  001176                  INC     $TMP3           ;ONE COMPARE FAILURE
7186  055514  023737  001172  001250  11$:    CMP     $TMP1,PMMR1     ;SEE IN MMR1 CHANGED
7187  055522  001402                          BEQ     12$             ;BRANCH IF MMR1 DIDN'T CHANGE
7188  055524  005237  001176                  INC     $TMP3           ;ANOTHER COMPARE FAILURE
7189  055530  023737  001170  001246  12$:    CMP     $TMP0,PMMR0     ;SEE IF MMR0 CHANGED
```

```
                                        BEQ     13$             ;BRANCH IF MMR0 DIDN'T CHANGE
                                        INC     $TMP3           ;ANOTHER COMPARE FAILURE
                                13$:    TST     $TMP3           ;WERE THERE ANY ERRORS ON THIS TEST
                                        BEQ     19$             ;BRANCH IF NO ERRORS
                                        ERROR   73              ;AT LEAST ONE M.M. REG CHANGED
                                19$:    BIC     #177776,MMR0   ;CLEAR ALL ERROR BITS IN MMR0
                                        MOV     #MMTRAP,MMVEC   ;PUT BACK REGULAR M.M. TRAP ROUTINE
                                        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

;;*********************************************************************
;*TEST 66        SUPERVISOR MODE, ABORT VECTOR FROM KERNEL SPACE
;*
;*      THIS TEST DOES AN ABORT FROM SUPERVISOR MODE.  THE VECTOR
;*      SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'
;*      FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.
;*
;*      THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE
;*      REACHED, IF SUPERVISOR MODE IS ENABLED PROPERLY ON 'SSRB',
;*      'SAPE', AND 'SAPB'.
;*
;;*********************************************************************
TST66:
                                        SCOPE
                                        MOV     #TST67,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                        TBITO                   ;MAKE SURE T-BIT IS OFF FOR THIS TEST
                                                                ;AND THE NEXT THREE (3) TESTS.
                                20$:    CLR     MMEXP           ;NOT EXPECTING ANY M.M. TRAPS YET
                                        MOV     #77400,UIPDR1   ;LOAD USER PAGE 1 NON-RESIDENT
                                        MOV     #77400,UIPDR2   ;LOAD USER PAGE 2 NON-RESIDENT
                                        MOV     #77400,UIPDR3   ;LOAD USER PAGE 3 NON-RESIDENT
                                        MOV     #77400,UIPDR7   ;LOAD USER PAGE 7 NON-RESIDENT
                                        MOV     #77406,R0       ;PAGE LENGTH-200 BLOCKS EXPAND UP
                                                                ;RESIDENT READ/WRITE
                                        MOV     R0,UIPDR0       ;LOAD USER PAGE 0
                                        MOV     R0,SIPDR0       ;LOAD SUPERVISOR PAGE 0
                                        MOV     R0,SIPDR1       ;LOAD SUPERVISOR PAGE 1
                                        MOV     R0,SIPDR2       ;LOAD SUPERVISOR PAGE 2
                                        MOV     R0,SIPDR3       ;LOAD SUPERVISOR PAGE 3
                                        MOV     R0,SIPDR7       ;LOAD SUPERVISOR PAGE 7
                                        MOV     #2,UIPAR0       ;MAP USER PAGE 0 TO 00200
                                        MOV     #177600,UIPAR7  ;MAP USER PAGE 7 TO I/O PAGE
                                        MOV     #1,SIPAR0       ;MAP SUPERVISOR PAGE 0 TO 000100
                                        MOV     #201,SIPAR1     ;MAP SUPERVISOR PAGE 1 TO 020100
                                        MOV     #401,SIPAR2     ;MAP SUPERVISOR PAGE 2 TO 040100
                                        MOV     #601,SIPAR3     ;MAP SUPERVISOR PAGE 3 TO 060100
                                        MOV     #177600,SIPAR7  ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
                                        MOV     #77403,SIPDR4   ;MAKE SUPERVISOR PAGE 4 NON-RESIDENT
                                        MOV     #1000,SIPAR4    ;MAP SUPERVISOR PAGE 4 TO 16K
                                        MOV     #SUPVEC,350     ;SUPERVISOR SPACE VECTOR
                                        MOV     #140,352        ;SUPERVISOR SPACE PSW = 140
                                        MOV     #USEVEC,450     ;USER SPACE VECTOR
                                        MOV     #000,452        ;USER SPACE PSW = 000
                                        MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
                                5$:     MOV     #40000,PSW      ;GO TO SUPERVISOR MODE.
```

```
                                                        ;THE NEXT INSTRUCTION EXECUTED IS AT
                                                        ;3$.  THE ADDRESS IS 100 OCTAL BYTES
                                                        ;GREATER THAN THE ADDRESS AT 1$.
056032  104074                  1$:     ERROR   74      ;DIDN'T GO TO SUPERVISOR MODE
056034  005037  177776                  CLR     PSW     ;GO BACK INTO KERNEL MODE
056040  000137  056102                  JMP     2$      ;GO TO EXIT OF TEST
056044  000000                          HALT            ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
056046  000000                          HALT            ;EVER BE REACHED
056050  000000                          HALT            ;EVER BE REACHED
056052  000000                          HALT            ;EVER BE REACHED
056054  000000                          HALT            ;EVER BE REACHED
056056  000000                          HALT            ;EVER BE REACHED
056060  000000                          HALT            ;EVER BE REACHED
056062  000000                          HALT            ;EVER BE REACHED
056064  000000                          HALT            ;EVER BE REACHED
056066  000000                          HALT            ;EVER BE REACHED
056070  000000                          HALT            ;EVER BE REACHED
056072  000000                          HALT            ;EVER BE REACHED
056074  000000                          HALT            ;EVER BE REACHED
056076  000000                          HALT            ;EVER BE REACHED
056100  000000                          HALT            ;EVER BE REACHED
056102  012737  025054  000250  2$:     MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP ROUTINE
056110  012737  055610  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
056116  000431                          BR      TST67           ;;BRANCH TO NEXT TEST
056120  000000                          HALT            ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
056122  000000                          HALT            ;EVER BE REACHED
056124  000000                          HALT            ;EVER BE REACHED
056126  000000                          HALT            ;EVER BE REACHED
056130  000000                          HALT            ;EVER BE REACHED
056132  012737  025304  000150  3$:     MOV     #KERVEC,<MMVEC-100>     ;SET UP KERNEL SPACE VECTOR
056140  012737  000340  000152          MOV     #340,<MMVEC+2-100>      ;KERNEL SPACE PSW = 340
056146  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
056150  013700  100000                  MOV     @#100000,R0             ;READ FROM PAGE 4, SUPERVISOR
056154  022737  100051  001146          CMP     #100051,<PMMR0-100>     ;EXPECTING NON-RESIDENT PAGE 4
                                                                        ;ABORT IN SUPERVISOR MODE
056162  001401                          BEQ     10$                     ;BRANCH IF CORRECT COND
056164  104075                          ERROR   75                      ;ABORT CONDITION INCORRECT
056166  042737  177776  177572  10$:    BIC     #177776,MMR0           ;CLEAR MMR0 FOR NEXT ABORT
056174  012737  000340  177776          MOV     #340,PSW        ;GO BACK INTO KERNEL MODE NOW
                                                                ;THE NEXT INSTRUCTION TO BE EXECUTED
                                                                ;IS AT LABEL 2$


                                ;;**********************************************************************
                                ;*TEST 67        M.M. ABORT DURING AN ODD ADDRESS ABORT SEQUENCE
                                ;*
                                ;*      THIS TEST VERIFIES BIT07 OF MMR0 (INSTRUCTION COMPLETE) AND
                                ;*      'SSRA PS RESTORE (1) H'.
                                ;*      THE TEST CAUSES AN 'ODD ADDRESS' ABORT THRU 'ERRVEC' WHICH
                                ;*      PUTS THE PROCESSOR INTO SUPERVISOR MODE.  THE SUPERVISOR
                                ;*      STACK POINTER IS AT 1104 AND ITS PAGE 0 IS 11 BLOCKS LONG,
                                ;*      SO WHEN THE OLD PS AND PC ARE PUSHED ON THE STACK A M.M.
                                ;*      PAGE LENGTH ABORT OCCURS.  THE OLD PS SHOULD BE RESTORED SO
                                ;*      THAT THE PROPER RECOVERY CAN BE MADE, AND THE M.M. ABORT
                                ;*      HAPPENS.
```

```
                                        ;*      MMR0  HAS PAGE LENGTH FAULT, SUPERVISOR I-SPACE, PAGE 0
                                        ;*      MMR1  HAS R6 DECREMENTED BY 2 TWICE
                                        ;*      MMR2  HAS 000004 (ADDRESS OF 'ERRVEC' WHERE M.M. ABORT OCCURRED)
                                        ;*
                                        ;;*****************************************************************
056202                                  TST67:
056202  000004                                  SCOPE
056204  012737  056566  001314                  MOV     #TST70,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
                                        .EQUIV  BIT4,TBIT               ;BIT 4 OF P.S. IS T-BIT TRAPPING BIT
056212  012737  040340  000006  20$:     MOV     #40340,ERRVEC+2 ;SET PRIORITY OF ERRVEC TO 7
                                                                        ;AND MAKE NEW MODE SUPERVISOR
056220  012737  000000  172240          MOV     #000,SIPAR0     ;MAP SUPERVISOR PAGE 0 TO PHYS. 0
056226  012737  000200  172242          MOV     #200,SIPAR1     ;MAP SUPERVISOR PAGE 1 TO 4K -8K
056234  012737  000400  172244          MOV     #400,SIPAR2     ;MAP SUPERVISOR PAGE 2 TO 8K - 12K
056242  012737  000600  172246          MOV     #600,SIPAR3     ;MAP SUPERVISOR PAGE 3 TO 12K - 16K
056250  012737  177600  172256          MOV     #177600,SIPAR7  ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
056256  012737  077406  172202          MOV     #77406,SIPDR1   ;MAKE SUPER PAGE 1 200 BLCKS, R/W
056264  012737  077406  172204          MOV     #77406,SIPDR2   ;MAKE SUPER PAGE 2 200 BLCKS, R/W
056272  012737  077406  172206          MOV     #77406,SIPDR3   ;MAKE SUPER PAGE 3 200 BLCKS, R/W
056300  012737  077406  172216          MOV     #77406,SIPDR7   ;MAKE SUPER PAGE 7 200 BLOCKS, R/W
056306  012737  056330  001110          MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
056314  012737  056362  000250          MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
056322  012737  004006  172200          MOV     #04006,SIPDR0   ;SUPERVISOR PAGE 0 EXPANDS UPWARD
                                                                        ;AND IS 11 BLOCKS LONG.
                                                                        ;ANY ADDRESS ABOVE 1076 WILL CAUSE A
                                                                        ;PAGE LENGTH FAULT.  THIS MEANS THAT
                                                                        ;WHEN THE ODD ADDRESS TRAP TRIES TO
                                                                        ;PUSH THE OLD PS ON THE SUPERVISOR STACK
                                                                        ;IT WILL GET A MEMORY MANAGEMENT ABORT.
056330  052737  040000  177776  1$:     BIS     #BIT14,PSW      ;GO TO SUPERVISOR MODE TO SET STK PTR.
056336  012706  001104                  MOV     #1104,SSP       ;SET THE STACK POINTER OUT OF LIMITS
056342  042737  040000  177776          BIC     #BIT14,PSW      ;GO BACK TO KERNEL MODE AND SET UP
                                                                        ;FOR ODD ADDRESS INSTRUCTION
056350  000230                          SPL     0               ;SET PRIORITY LEVEL TO 0
056352  000277                          SCC                     ;SET ALL CONDITION CODES
056354  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
056356  013701  000001                  MOV     @#000001,R1     ;TRY TO READ ADDRESS 1 INTO R1
                                                                        ;THIS WILL ABORT TO 4, AND THE PS
                                                                        ;AT ADDRESS 6 WILL FORCE SUPERVISOR.
                                                                        ;THEN WHILE PUSHING THE PS ON THE STACK
                                                                        ;YOU SHOULD GET A MEMORY MANAGEMENT
                                                                        ;ABORT AND GO TO 10$.  THE PS SHOULD
                                                                        ;HAVE BEEN RESTORED AND WILL NOW BE ON
                                                                        ;THE KERNEL STACK.
056362  016601  000002          10$:    MOV     2(KSP),R1       ;COPY PS ON STACK INTO R1
056366  012716  056374                  MOV     #16$,(KSP)      ;SET RETURN PC TO 16$
056372  000006                          RTT                     ;RETURN TO 16$ WITH T-BIT INTACT
056374  042701  000020          16$:    BIC     #TBIT,R1        ;CLEAR T BIT IN R1 IF ON THIS PASS
056400  005037  001176                  CLR     $TMP3           ;THIS IS THE ERROR COUNTER FLAG
056404  122701  000017                  CMPB    #17,R1          ;THE PROCESSOR STATUS THAT SHOULD
                                                                        ;BE ON THE STACK IS THE ONE WITH ALL
                                                                        ;OF THE CONDITION CODES SET.
056410  001402                          BEQ     11$             ;BRANCH IF PS IS CORRECT
056412  005237  001176                  INC     $TMP3           ;WRONG PS IS ON STACK
```

```
7358  056416  013737  177572  001246  11$:    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR COMPARE
7359  056424  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1, IT SHOULD BE 173366
7360  056432  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2, IT SHOULD EQUAL #10
7361  056440  022737  040241  001246          CMP     #040241,PMMR0   ;SHOULD HAVE PAGE LENGTH FAULT,
7362                                                                  ;COMPLETED, PAGE 0, SUPERVISOR I-SPACE.
7363  056446  001402                          BEQ     12$             ;BRANCH IF CORRECT CONDITION
7364  056450  005237  001176                  INC     $TMP3           ;WRONG ABORT CONDITION
7365  056454  022737  173366  001250  12$:    CMP     #173366,PMMR1   ;R6 DECREMENTED TWICE, DURING ABORTED
7366                                                                  ;PUSHES TO SUPERVISOR STACK
7367  056462  001402                          BEQ     13$             ;BRANCH IF MMR1 IS CORRECT
7368  056464  005237  001176                  INC     $TMP3           ;MMR1 IS NOT CORRECT
7369  056470  022737  000004  001252  13$:    CMP     #4,PMMR2        ;SEE IF MMR2 EQUALS ADDRS.000004
7370  056476  001402                          BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
7371  056500  005237  001176                  INC     $TMP3           ;MMR2 HAS THE WRONG ADDRESS
7372  056504  005737  001176          14$:    TST     $TMP3           ;DID ANY OF THE COMPARES FAIL?
7373  056510  001401                          BEQ     15$             ;BRANCH IF ALL COMPARES SUCCEEDED
7374  056512  104070                          ERROR   70              ;AT LEAST ONE COMPARE FAILED
7375  056514  000237          15$:    SPL     7                       ;NORMAL PRIORITY IS 7
7376  056516  052737  040000  177776          BIS     #BIT14,PSW      ;GO TO SUPERVISOR MODE TO RESET PTR
7377  056524  012706  000700                  MOV     #700,SSP        ;RESTORE SUPER STK PTR TO 700
7378  056530  042737  040000  177776          BIC     #BIT14,PSW      ;RETURN TO KERNEL MODE
7379  056536  012737  000340  000006          MOV     #340,ERRVEC+2   ;RESTORE CORRECT PS TO ERROR VECTOR
7380  056544  012737  056212  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
7381  056552  042737  177776  177572          BIC     #177776,MMR0    ;CLEAR ERROR CONDITION IN MMR0
7382  056560  012737  025054  000250          MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. VECTOR
7383
7384
7385                                  ;;*****************************************************************
7386                                  ;*TEST 70       USER MODE, ABORT VECTOR FROM KERNEL SPACE
7387                                  ;*
7388                                  ;*      THIS TEST DOES AN ABORT FROM USER MODE.  THE VECTOR
7389                                  ;*      SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'
7390                                  ;*      FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.
7391                                  ;*
7392                                  ;*      THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE
7393                                  ;*      REACHED, IF USER MODE IS ENABLED PROPERLY ON 'SSRB', 'SAPE',
7394                                  ;*      'SAPB', 'SAPC', AND 'SAPF'.
7395                                  ;*
7396                                  ;*      IT SHOULD BE NOTED THAT IF THIS TEST CODE IS EXECUTED IN SINGLE
7397                                  ;*      INSTRUCTION, THE ABORT WILL PUSH THE PS & PC ONTO THE SUPERVISOR
7398                                  ;*      STACK INSTEAD OF THE KERNEL STACK.  THIS IS DUE TO A FLAW IN
7399                                  ;*      THE CPU ROM AND IS A CARRY OVER FROM THE PDP-11/45.  IF YOU NEED
7400                                  ;*      TO SINGLE INSTRUCTION THIS TEST, SINGLE BUS CYCLE THRU THE ABORT
7401                                  ;*      SEQUENCE FOR PROPER OPERATION OF THE STACKS.
7402                                  ;*
7403                                  ;;*****************************************************************
7404  056566                  TST70:
7405  056566  000004                  SCOPE
7406  056570  012737  057170  001314          MOV     #TST71,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
7407                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
7408  056576  005037  001224          20$:    CLR     MMEXP           ;NOT EXPECTING ANY M.M. TRAPS YET
7409  056602  012737  077400  172202          MOV     #77400,SIPDR1   ;LOAD SUPERVISOR PAGE 1 NON-RESIDENT
7410  056610  012737  077400  172204          MOV     #77400,SIPDR2   ;LOAD SUPERVISOR PAGE 2 NON-RESIDENT
7411  056616  012737  077400  172206          MOV     #77400,SIPDR3   ;LOAD SUPERVISOR PAGE 3 NON-RESIDENT
7412  056624  012737  077400  172216          MOV     #77400,SIPDR7   ;LOAD SUPERVISOR PAGE 7 NON-RESIDENT
7413  056632  012700  077406                  MOV     #77406,R0       ;PAGE LENGTH-200 BLOCKS EXPAND UP
```

```
                                                                ;RESIDENT READ/WRITE
056636  010037  172200          MOV     R0,SIPDR0             ;LOAD SUPERVISOR PAGE 0
056642  010037  177600          MOV     R0,UIPDR0             ;LOAD USER PAGE 0
056646  010037  177602          MOV     R0,UIPDR1             ;LOAD USER PAGE 1
056652  010037  177604          MOV     R0,UIPDR2             ;LOAD USER PAGE 2
056656  010037  177606          MOV     R0,UIPDR3             ;LOAD USER PAGE 3
056662  010037  177616          MOV     R0,UIPDR7             ;LOAD USER PAGE 7
056666  012737  000002  172240  MOV     #2,SIPAR0             ;MAP SUPERVISOR PAGE 0 TO 00200
056674  012737  177600  172256  MOV     #177600,SIPAR7        ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
056702  012737  000001  177640  MOV     #1,UIPAR0             ;MAP USER PAGE 0 TO 00100
056710  012737  000201  177642  MOV     #201,UIPAR1           ;MAP USER PAGE 1 TO 20100
056716  012737  000401  177644  MOV     #401,UIPAR2           ;MAP USER PAGE 2 TO 40100
056724  012737  000601  177646  MOV     #601,UIPAR3           ;MAP USER PAGE 3 TO 60100
056732  012737  177600  177656  MOV     #177600,UIPAR7        ;MAP USER PAGE 7 TO I/O PAGE
056740  012737  077400  177612  MOV     #77400,UIPDR5         ;MAKE USER PAGE 5 NON-RESIDENT
056746  012737  001000  177652  MOV     #1000,UIPAR5          ;MAP USER PAGE 5 TO 16K
056754  012737  025334  000450  MOV     #SUPVEC,450           ;SUPERVISOR SPACE VECTOR
056762  012737  000140  000452  MOV     #140,452              ;SUPERVISOR SPACE PSW = 140
056770  012737  025356  000350  MOV     #USEVEC,350           ;USER SPACE VECTOR
056776  012737  000000  000352  MOV     #000,352              ;USER SPACE PSW = 000
057004  012737  057012  001110  MOV     #5$,$LPERR            ;SET LOOP ON ERROR POINTER TO 5$
057012  012737  140000  177776  5$:     MOV     #140000,PSW   ;GO TO USER MODE.
                                                                ;THE NEXT INSTRUCTION EXECUTED IS AT
                                                                ;3$.  THE ADDRESS IS 100 OCTAL BYTES
                                                                ;GREATER THAN THE ADDRESS AT 1$.
057020  104076                  1$:     ERROR   76            ;DIDN'T GO TO USER MODE
057022  005037  177776          CLR     PSW                   ;GO BACK INTO KERNEL MODE
057026  000137  057070          JMP     2$                    ;GO TO EXIT OF TEST
057032  000000                  HALT                          ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
057034  000000                  HALT                          ;EVER BE REACHED
057036  000000                  HALT                          ;EVER BE REACHED
057040  000000                  HALT                          ;EVER BE REACHED
057042  000000                  HALT                          ;EVER BE REACHED
057044  000000                  HALT                          ;EVER BE REACHED
057046  000000                  HALT                          ;EVER BE REACHED
057050  000000                  HALT                          ;EVER BE REACHED
057052  000000                  HALT                          ;EVER BE REACHED
057054  000000                  HALT                          ;EVER BE REACHED
057056  000000                  HALT                          ;EVER BE REACHED
057060  000000                  HALT                          ;EVER BE REACHED
057062  000000                  HALT                          ;EVER BE REACHED
057064  000000                  HALT                          ;EVER BE REACHED
057066  000000                  HALT                          ;EVER BE REACHED
057070  012737  025054  000250  2$:     MOV     #MMTRAP,MMVEC ;RESTORE NORMAL M.M. TRAP ROUTINE
057076  012737  056576  001110          MOV     #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
057104  000431                  BR      TST71                 ;;BRANCH TO NEXT TEST
057106  000000                  HALT                          ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
057110  000000                  HALT                          ;EVER BE REACHED
057112  000000                  HALT                          ;EVER BE REACHED
057114  000000                  HALT                          ;EVER BE REACHED
057116  000000                  HALT                          ;EVER BE REACHED
057120  012737  025304  000150  3$:     MOV     #KERVEC,<MMVEC-100>     ;SET UP KERNEL SPACE VECTOR
057126  012737  000340  000152          MOV     #340,<MMVEC+2-100>      ;KERNEL SPACE PSW = 340
057134  000240                  NOP                           ;THIS IS A SYNC POINT FOR SCOPING
057136  013700  120000          MOV     @#120000,R0           ;READ FROM PAGE 5, USER SPACE
057142  022737  100153  001146  CMP     #100153,<PMMR0-100>   ;EXPECTING NON-RESIDENT PAGE 5
```

```
                                                        ;ABORT IN USER MODE I-SPACE
057150  001401                      BEQ     10$           ;BRANCH IF CORRECT COND
057152  104077                      ERROR   77            ;ABORT CONDITION INCORRECT
057154  042737  177776  177572  10$:BIC     #177776,MMR0  ;CLEAR MMR0 FOR NEXT TEST
057162  012737  000340  177776      MOV     #340,PSW    ;GO BACK INTO KERNEL MODE NOW
                                                        ;THE NEXT INSTRUCTION TO BE EXECUTED
                                                        ;IS AT LABEL 2$


                                ;;**********************************************************************
                                ;*TEST 71       COUNT PATTERN THRU MMR2, TO TEST ALL BITS
                                ;*
                                ;*      THIS TEST SETS UP ALL USER I-SPACE PAGES TO BE NON-RESIDENT
                                ;*      AND THEN TRIES ALL POSSIBLE VIRTUAL ADDRESSES AS A PROCESSOR
                                ;*      COUNTER IN USER MODE.  EVERY INSTRUCTION FETCH WILL ABORT,
                                ;*      NON RESIDENT AND THE CONTENTS OF MMR2 IS TESTED ALONG
                                ;*      WITH BITS <06:01> OF MMR0.
                                ;*
                                ;;**********************************************************************
057170                          TST71:
057170  000004                      SCOPE
057172  012737  057434  001314      MOV     #TST72,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
057200  012737  000002  001204      MOV     #2,$TIMES       ;;DO 2 ITERATIONS
057206  012737  077400  177600  20$:MOV     #77400,UIPDR0   ;MAP USER PAGE 0 NON-RESIDENT
057214  012737  077400  177602      MOV     #77400,UIPDR1   ;MAP USER PAGE 1 NON-RESIDENT
057222  012737  077400  177604      MOV     #77400,UIPDR2   ;MAP USER PAGE 2 NON-RESIDENT
057230  012737  077400  177606      MOV     #77400,UIPDR3   ;MAP USER PAGE 3 NON-RESIDENT
057236  012737  077400  177610      MOV     #77400,UIPDR4   ;MAP USER PAGE 4 NON-RESIDENT
057244  012737  077400  177612      MOV     #77400,UIPDR5   ;MAP USER PAGE 5 NON-RESIDENT
057252  012737  077400  177614      MOV     #77400,UIPDR6   ;MAP USER PAGE 6 NON-RESIDENT
057260  012737  077400  177616      MOV     #77400,UIPDR7   ;MAP USER PAGE 7 NON-RESIDENT
057266  012737  057316  001110      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
057274  012700  140000              MOV     #140000,R0      ;THIS WILL FORCE USER MODE WHEN USED
                                                        ;AS A PROCESSOR STATUS
057300  005001                      CLR     R1              ;R1 HOLDS USER VIRTUAL PC
057302  012737  057326  000250      MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
057310  012737  000340  000252      MOV     #340,MMVEC+2    ;MAKE SURE TRAP TAKES YOU TO KERNEL
057316  010046              2$:     MOV     R0,-(KSP)       ;PUSH USER PS ON STACK
057320  010146                      MOV     R1,-(KSP)       ;PUSH USER'S VIRTUAL PC ON STACK
057322  000240                      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
057324  000002                      RTI                     ;RETURN TO USER MODE
                                                        ;THE FIRST INSTRUCTION FETCH WILL
                                                        ;CAUSE A NON-RESIDENT ABORT AND LOCK
                                                        ;MMR2 SO THAT ALL BITS CAN BE CHECKED.


057326  062706  000004      10$:    ADD     #4,KSP          ;CLEAN UP STACK FOR NEXT TIME
057332  013737  177576  001252      MOV     MMR2,PMMR2      ;READ MMR2 TO TEMP LOCATION
057340  013737  177572  001246      MOV     MMR0,PMMR0      ;READ MMR0 TO TEMP LOCATION
057346  020137  001252              CMP     R1,PMMR2        ;SEE IF MMR2 LOCKED CORRECT V. A.
057352  001401                      BEQ     11$             ;BRANCH IF MMR2 HAS RIGHT V.A.
057354  104100                      ERROR   100             ;WRONG V.A.
057356  005002              11$:    CLR     R2              ;R2 WILL GET THE VIRTUAL PAGE NO.
057360  010103                      MOV     R1,R3           ;COPY VIRTUAL ADDRESS INTO R3
```

```
                                                ;COMBINED LEFT SHIFT <R2,R3> 3 BITS
                                                ;ADJUST PAGE NUMBER
                                                ;SET OTHER EXPECTED BITS IN MMR0
                                                ;SEE IF MMR0 RECORDED CORRECT PAGE NO.
                                                ;BRANCH IF PAGE NUMBER WAS CORRECT
                                                ;WRONG PAGE NO. IN MMR0
                                                ;CLEAR ALL ERROR BITS IN MMR0
                                                ;TRY NEXT VIRTUAL ADDRESS
                                                ;BRANCH IF NOT ALL DONE
                                                ;PUT BACK REGULAR M.M. TRAP ROUTINE
                                                ;SET LOOP POINTER TO START OF TEST
```

```
                                .SBTTL  ************ ENTRY POINT 6 --- STARTING ADDRESS 224 ************
                                .SBTTL  ********* D-SPACE TESTS, CORRECT TIMING OF I & D SPACE *********
                                ;*
                                ;*      THIS GROUP OF TESTS CHECKS THE PROPER ENABLING OF D-SPACE.
                                ;*      IT TESTS THAT I-SPACE IS FORCED DURING INSTRUCTION FETCHES AND
                                ;*      ADDRESS, INDEX, OR OPERAND FETCHES IF THE REGISTER FIELD IS 7.
                                ;*      IT ALSO CHECKS THAT TRAPS PICK UP THE VECTOR FROM D-SPACE, IF
                                ;*      IT IS ENABLED.
                                ;*
                                ;*      THROUGHOUT THIS AREA OF TESTING D-SPACE PAGES 2 & 3
                                ;*      ARE MAPPED NON-RESIDENT, AND I-SPACE PAGE 4 IS ALSO MAPPED
                                ;*      NON-RESIDENT.  ALL OTHER PAGES IN BOTH I & D SPACE ARE
                                ;*      MAPPED RESIDENT, 4K, READ/WRITE.  IF ANY ABORTS SHOULD OCCUR
                                ;*      DURING THESE TESTS THEY WILL VECTOR TO 'NODSPAC''.  IF THE
                                ;*      OFFENDING PAGE IS 2 OR 3 THEN THE FAULT IS THAT I-SPACE WASN'T
                                ;*      FORCED WHEN IT SHOULD HAVE BEEN, BUT IF THE PAGE IS 4 THEN
                                ;*      THE FAULT IS THAT I-SPACE WAS FORCED WHEN IT SHOULD NOT HAVE BEEN.
                                ;*
```

```
                                ;;*********************************************************************
                                ;*TEST 72       ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS FORCED
                                ;*
                                ;*      THIS TEST SHOWS THAT I-SPACE IS FORCED BY EITHER 'SSRB I
                                ;*      SPACEA L' OR SSRB I SPACEB L' DURING THE PROPER TIMES
                                ;*
                                ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
                                ;*      ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC''.  THIS SUBROUTINE
                                ;*      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
                                ;*
                                ;;*********************************************************************
                                TST72:
                                        SCOPE
                                        MOV     #TST73,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                        TBITR                   ;RESTORE THE T-BIT TO ITS CONDITION
                                                                ;BEFORE THE LAST FOUR TESTS.
                                ENTPT6:
                                        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
                                        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
                                        MOV     #72,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
                                        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
                                        MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
```

```
 7582  057504  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
 7583  057512  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
 7584  057520  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
 7585  057526  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
 7586  057534  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
 7587  057542  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
 7588  057550  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
 7589  057556  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
 7590  057564  012737  177600  172356          MOV     #177600,KIPAR7  ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
 7591  057572  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
 7592  057600  012737  000020  172516          MOV     #BIT4,MMR3      ;ENABLE 22-BIT RELOCATION IF NOT ON
 7593  057606  012737  077400  172310  20$:    MOV     #77400,KIPDR4   ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 7594  057614  012737  077406  172320          MOV     #77406,KDPDR0   ;MAKE KERNEL D PAGE 0 200 BLOCKS R/W
 7595  057622  012737  077406  172322          MOV     #77406,KDPDR1   ;MAKE KERNEL D PAGE 1 200 BLOCKS R/W
 7596  057630  012737  077406  172330          MOV     #77406,KDPDR4   ;MAKE KERNEL D PAGE 4 200 BLOCKS R/W
 7597  057636  012737  077400  172324          MOV     #77400,KDPDR2   ;MAKE KERNEL D PAGE 2 NON-RESIDENT
 7598  057644  012737  077400  172326          MOV     #77400,KDPDR3   ;MAKE KERNEL D PAGE 3 NON-RESIDENT
 7599  057652  012737  000000  172360          MOV     #000,KDPAR0     ;MAP KERNEL D PAGE 0 TO PHYSICAL 0
 7600  057660  012737  000200  172362          MOV     #200,KDPAR1     ;MAP KERNEL D PAGE 1 TO 4K - 8K
 7601  057666  012737  001000  172370          MOV     #1000,KDPAR4    ;MAP KERNEL D PAGE 4 TO 16K
 7602  057674  012737  077406  172336          MOV     #77406,KDPDR7   ;MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
 7603  057702  012737  177600  172376          MOV     #177600,KDPAR7  ;MAP KERNEL D PAGE 7 TO I/O PAGE
 7604  057710  012704  172516                  MOV     #MMR3,R4        ;PUT ADDRS OF MMR3 IN R4
 7605  057714  012705  000004                  MOV     #BIT2,R5        ;PUT KERNEL D-SPACE ENABLE BIT INTO R5
 7606  057720  012700  000002                  MOV     #2,R0           ;LOAD A TWO INTO R0
 7607  057724  012737  025174  000250          MOV     #NODSPAC,MMVEC  ;SET M.M. VECTOR TO D-SPACE SERVICE ROUTINE
 7608  057732  012737  057742  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
 7609  057740  050514                          BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
 7610                                  ;;
 7611                                  ;*      TEST THAT 'ROM OUT09' COMES UP ON INSTRUCTION FETCHES,
 7612                                  ;*      FORCING I-SPACE.
 7613                                  ;;
 7614  057742  000400                  10$:    BR      1$              ;BRANCH, USE FET.00
 7615  057744  000244                  1$:     CLZ                     ;CLEAR ZERO BIT IN PROCESSOR STATUS
 7616  057746  001776                          BEQ     1$              ;NO BRANCH, USE FET.13
 7617  057750  000264                  2$:     SEZ                     ;SET ZERO BIT IN PROC. STATUS
 7618  057752  001376                          BNE     2$              ;NO BRANCH, USE FET.12
 7619  057754  000270                  3$:     SEN                     ;SET NEGATIVE BIT IN PROC. STATUS
 7620  057756  100376                          BPL     3$              ;NO BRANCH, USE FET.11
 7621  057760  000237                  4$:     SPL     7               ;SET PRIOR TO 7: USE SPL.10, GOTO FET.10
 7622  057762  005700                          TST     R0              ;USE TST.10, GOTO FET.10
 7623  057764  077003                          SOB     R0,4$           ;BRANCH UNTIL R0 IS ZERO:
                                                                       ;USE SOB.20, GOTO FET.10
 7624
 7625  057766  073002                          ASHC    R2,R0           ;COMBINED SHIFT, SHIFT COUNT ZERO
                                                                       ;USE ASC.80, GOTO FET.10
 7626
 7627  057770  005200                          INC     R0              ;MAKE R0 POSITIVE ONE
 7628  057772  073002                          ASHC    R2,R0           ;LEFT COMBINED SHIFT ONE PLACE
                                                                       ;USE ASC.61, GOTO FET.10
 7629
 7630  057774  005300                          DEC     R0              ;MAKE R0 EQUAL ZERO
 7631  057776  005300                          DEC     R0              ;MAKE R0 NEGATIVE ONE
 7632  060000  073002                          ASHC    R2,R0           ;RIGHT COMBINED SHIFT ONE PLACE
                                                                       ;USE ASC.60, GOTO FET.10
 7633
 7634  060002  072001                          ASH     R1,R0           ;RIGHT SHIFT ONE PLACE
                                                                       ;GOTO FET.07
 7635
 7636  060004  005200                          INC     R0              ;MAKE R0 EQUAL ZERO
 7637  060006  005200                          INC     R0              ;MAKE R0 POSITIVE ONE
```

```
7638  060010  072001                        ASH     R1,R0           ;LEFT SHIFT ONE PLACE
7639                                                                  ;GO TO FET.05
7640  060012  070001                        MUL     R1,R0           ;MULTIPLY R1 X R0
7641                                                                  ;USE MUL.60, GOTO FET.10
7642  060014  005000                        CLR     R0              ;MAKE SURE R0 IS ZERO
7643  060016  071001                        DIV     R1,R0           ;DIVISOR IS ZERO, GO TO FET.04
7644  060020  040514                        BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7645                                ;;
7646                                ;*      TEST THAT 'ROM OUT13'' COMES UP WHEN SRCM = 1, 2, OR 3.
7647                                ;*      THIS IS ANDED WITH SRCF = 7
7648                                ;;
7649  060022  012737  060032  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
7650  060030  050514                        BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
7651  060032  011700                11$:    MOV     (PC),R0         ;USE S13.00, SOURCE MODE IS 1
7652  060034  012700  000000                MOV     #0,R0           ;USE S13.01, SOURCE MODE IS 2
7653  060040  013700  100000                MOV     @#100000,R0     ;USE S13.01, SOURCE MODE IS 3
7654  060044  040514                        BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7655                                ;;
7656                                ;*      TEST THAT 'ROM OUT14'' COMES UP WHEN DSTM = 1, OR 2.
7657                                ;*      THIS IS ANDED WITH DSTF = 7 AND NOT(MTP + MFP).
7658                                ;;
7659  060046  012737  060056  001110        MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
7660  060054  050514                        BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
7661  060056  005717                12$:    TST     (PC)            ;USE D12.00, GOTO D12.10
7662                                                                  ;(INST. IS DAC * DM1)
7663  060060  005727  000000                TST     #0              ;USE D12.01, GOTO D12.10
7664                                                                  ;(INST IS DAC * DM2)
7665  060064  022717  000240                CMP     #240,(PC)       ;USE D12.80, GOTO D12.60
7666                                                                  ;(INST IS BIN * DM1)
7667  060070  000240                        NOP                     ;THIS SHOULD BE COMPARED WITH IMMEDIATE
7668                                                                  ;DATA IN ABOVE COMPARE INSTRUCTION
7669  060072  040514                        BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7670                                ;;
7671                                ;*      TEST THAT 'ROM OUT15'' COMES UP WHEN DSTM = 3.
7672                                ;*      THIS IS ANDED WITH DSTF = 7.
7673                                ;;
7674  060074  012737  060104  001110        MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
7675  060102  050514                        BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
7676  060104  005037  100000        13$:    CLR     @#100000        ;CLEAR LOCATION 100000
7677                                                                  ;USE D30.00, GOTO D30.10
7678  060110  012737  000000  100000        MOV     #0,@#100000     ;LOAD ZERO TO LOCATION 100000
7679                                                                  ;USE D30.80, GOTO D30.10
7680  060116  112737  000000  100001        MOVB    #0,@#100001     ;LOAD ZERO INTO UPPER BYTE OF 100000
7681                                                                  ;USE D30.90, GOTO D30.10
7682  060124  040514                        BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7683                                ;;
7684                                ;*      TEST THAT 'ROM OUT09'' FORCES I-SPACE DURING INDEX WORD
7685                                ;*      FETCHES.
7686                                ;;
7687  060126  012737  060136  001110        MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
7688  060134  050514                        BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
7689  060136  012700  000000        14$:    MOV     #0,R0           ;BIN * DM0, USE D00.90, GOTO FET.10
7690  060142  010001                        MOV     R0,R1           ;BIN * DM0 * SM0, USE EXC.80, GOTO FET.10
7691  060144  012767  000001  017626        MOV     #1,<77772-.>(PC)        ;LOAD 1 INTO ADR 100000
7692                                                                  ;DSTM = 6, DSTF = 7
7693                                                                  ;USE D67.80, AND D67.00
```

```
7694  060152  112777  000000  017620          MOVB    #0,@100000      ;LOAD ZERO INTO ADR 000001
7695                                                                  ;DSTM = 7, DSTF = 7
7696                                                                  ;USE D67.90 AND D67.01
7697  060160  016700  017614                  MOV     <77774-.>(PC),R0        ;READ FROM ADRS 100000
7698                                                                  ;SRCM = 6, SRCF = 7
7699                                                                  ;USE S67.00
7700  060164  040514                          BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7701                                  ;:
7702                                  ;*      TEST 'ROM OUT11'' ANDED WITH (PREV=I)
7703                                  ;:
7704  060166  012737  060176  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
7705  060174  050514                          BIS     R5,(R4)         ;ENABLE D-SPACE MAPPING IN KERNEL MODE
7706  060176  012737  000000  100000  15$:     MOV     #0,@#100000     ;CLEAR LOCATION 100000
7707  060204  005437  100000                  NEG     @#100000        ;NEGATE ZERO, USE NEG.20, GOTO EXC.10
7708  060210  105437  100001                  NEGB    @#100001        ;NEGATE UPPER BYTE OF 100000
7709                                                                  ;USE SHR.10, GOTO EXC.10
7710  060214  005537  100000                  ADC     @#100000        ;ADD CARRY BIT TO ADRS 100000
7711                                                                  ;USE EXC.00, GOTO EXC.10
7712  060220  040514                  19$:     BIC     R5,(R4)         ;DISABLE KERNEL D-SPACE MAPPING
7713  060222  012737  057606  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
7714
7715
7716                                  ;;**************************************************************
7717                                  ;*TEST 73       ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS NOT FORCED
7718                                  ;*
7719                                  ;*      THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER
7720                                  ;*      FIELD IS NOT 7 BUT THE OTHER CONDITIONS ARE STILL MET.
7721                                  ;*
7722                                  ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
7723                                  ;*      ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'.  THIS SUBROUTINE
7724                                  ;*      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
7725                                  ;*
7726                                  ;;**************************************************************
7727  060230                          TST73:
7728  060230  000004                          SCOPE
7729  060232  012737  060532  001314          MOV     #TST74,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
7730                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
7731  060240  012737  077406  172310  20$:     MOV     #77406,KIPDR4   ;MAKE KERNEL I PAGE 4 200 BLOCKS, R/W
7732  060246  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
7733  060254  012737  077406  172320          MOV     #77406,KDPDR0   ;MAKE KERNEL D PAGE 0 200 BLOCKS R/W
7734  060262  012737  077406  172322          MOV     #77406,KDPDR1   ;MAKE KERNEL D PAGE 1 200 BLOCKS R/W
7735  060270  012737  077406  172330          MOV     #77406,KDPDR4   ;MAKE KERNEL D PAGE 4 200 BLOCKS R/W
7736  060276  012737  077400  172324          MOV     #77400,KDPDR2   ;MAKE KERNEL D PAGE 2 NON-RESIDENT
7737  060304  012737  077400  172326          MOV     #77400,KDPDR3   ;MAKE KERNEL D PAGE 3 NON-RESIDENT
7738  060312  012737  000000  172360          MOV     #000,KDPAR0     ;MAP KERNEL D PAGE 0 TO PHYSICAL 0
7739  060320  012737  001000  172370          MOV     #1000,KDPAR4    ;MAP KERNEL D PAGE 4 TO 16K
7740  060326  012737  077406  172336          MOV     #77406,KDPDR7   ;MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
7741  060334  012737  177600  172376          MOV     #177600,KDPAR7  ;MAP KERNEL D PAGE 7 TO I/O PAGE
7742  060342  012704  172516                  MOV     #MMR3,R4        ;PUT ADDRS OF MMR3 IN R4
7743  060346  012705  000004                  MOV     #BIT2,R5        ;PUT KERNEL D-SPACE ENABLE BIT INTO R5
7744  060352  012700  100000                  MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
7745  060356  012737  100000  100000          MOV     #100000,@#100000 ;LOAD ADDRESS 100000 WITH 100000
7746  060364  012737  100000  100002          MOV     #100000,@#100002 ;LOAD ADDRESS 100002 WITH 100000
7747  060372  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
7748  060376  012737  060404  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
7749                                  ;:
```

```
                                        ;*      TEST THAT 'ROM OUT13'' COMES UP WHEN SRCM = 1, 2, OR 3.
                                        ;*      IN THIS CASE SRCF = -7.
                                        ;:
060404  012700  100000          11$:    MOV     #100000,R0          ;LOAD ADDRESS 100000 INTO R0
060410  050514                          BIS     R5,(R4)             ;ENABLE D-SPACE MAPPING IN KERNEL MODE
060412  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
060414  011001                          MOV     (R0),R1             ;USE S13.00, SOURCE MODE IS 1
060416  012001                          MOV     (R0)+,R1            ;USE S13.01, SOURCE MODE IS 2
060420  013001                          MOV     @(R0)+,R1           ;USE S13.01, SOURCE MODE IS 3
060422  040514                          BIC     R5,(R4)             ;DISABLE KERNEL D-SPACE MAPPING
                                        ;:
                                        ;*      TEST THAT 'ROM OUT14'' COMES UP WHEN DSTM = 1, OR 2.
                                        ;*      THIS IS ANDED WITH DSTF = -7 AND NOT(MTP + MFP).
                                        ;:
060424  012737  060432  001110          MOV     #12$,$LPERR         ;SET LOOP ON ERROR POINTER TO 12$
060432  012700  100000          12$:    MOV     #100000,R0          ;LOAD ADDRESS 100000 INTO R0
060436  050514                          BIS     R5,(R4)             ;ENABLE D-SPACE MAPPING IN KERNEL MODE
060440  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
060442  005710                          TST     (R0)                ;USE D12.00, GOTO D12.10
                                                                    ;(INST. IS DAC * DM1)
060444  005720                          TST     (R0)+               ;USE D12.01, GOTO D12.10
                                                                    ;(INST IS DAC * DM2)
060446  021010                          CMP     (R0),(R0)           ;USE D12.80, GOTO D12.60
                                                                    ;(INST IS BIN * DM1)
060450  122020                          CMPB    (R0)+,(R0)+         ;USE D12.90, GOTO D12.60
                                                                    ;(INST IS BIN * DM2)
060452  040514                          BIC     R5,(R4)             ;DISABLE KERNEL D-SPACE MAPPING
                                        ;:
                                        ;*      TEST THAT 'ROM OUT15'' COMES UP WHEN DSTM = 3.
                                        ;*      THIS IS ANDED WITH DSTF = -7.
                                        ;:
060454  012737  060462  001110          MOV     #13$,$LPERR         ;SET LOOP ON ERROR POINTER TO 13$
060462  112737  000006  172310  13$:    MOVB    #006,KIPDR4         ;MAKE KERNEL I-SPACE PAGE 4 RESIDENT
060470  012700  100000                  MOV     #100000,R0          ;LOAD ADDRESS 100000 INTO R0
060474  105037  172310                  CLRB    KIPDR4              ;MAKE KERNEL I PAGE 4 NON-RESIDENT
060500  050514                          BIS     R5,(R4)             ;ENABLE D-SPACE MAPPING IN KERNEL MODE
060502  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
060504  005030                          CLR     @(R0)+              ;USE D30.00, GOTO D30.10
060506  012730  100000                  MOV     #100000,@(R0)+      ;USE D30.80, GOTO D30.10
060512  012710  100001                  MOV     #100001,(R0)        ;LOAD ODD ADDRESS INTO LOCATION OF R0
060516  112730  000200                  MOVB    #200,@(R0)+         ;USE D30.90, GOTO D30.10
060522  040514                  19$:    BIC     R5,(R4)             ;DISABLE KERNEL D-SPACE MAPPING
060524  012737  060240  001110          MOV     #20$,$LPERR         ;SET LOOP POINTER TO START OF TEST


                                        ;;*****************************************************************
                                        ;*TEST 74       PROPER ENABLING OF SUPERVISOR D-SPACE
                                        ;*
                                        ;*      THIS TEST SHOWS THE PROPER FUNCTIONING OF SUPERVISOR D-SPACE.
                                        ;*      BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
                                        ;*      DURING THIS TEST FORCING I-SPACE.
                                        ;*
                                        ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
                                        ;*      ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC''.  THIS SUBROUTINE
                                        ;*      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
```

```
                                        ;*
                                        ;;*****************************************************************
060532                                  TST74:
                                                SCOPE
060532  000004
060534  012737  061204  001314                  MOV     #TST75,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
060542  012700  077406              20$:        MOV     #77406,R0       ;MAKE THE NEXT FEW PAGES 4K, R/W, UP
060546  010037  172200                          MOV     R0,SIPDR0       ;SUPERVISOR I-SPACE PAGE 0
060552  010037  172202                          MOV     R0,SIPDR1       ;SUPERVISOR I-SPACE PAGE 1
060556  010037  172204                          MOV     R0,SIPDR2       ;SUPERVISOR I-SPACE PAGE 2
060562  010037  172206                          MOV     R0,SIPDR3       ;SUPERVISOR I-SPACE PAGE 3
060566  010037  172210                          MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
060572  010037  172216                          MOV     R0,SIPDR7       ;SUPERVISOR I-SPACE PAGE 7
060576  010037  172220                          MOV     R0,SDPDR0       ;SUPERVISOR D-SPACE PAGE 0
060602  010037  172222                          MOV     R0,SDPDR1       ;SUPERVISOR D-SPACE PAGE 1
060606  010037  172230                          MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
060612  010037  172236                          MOV     R0,SDPDR7       ;SUPERVISOR D-SPACE PAGE 7
060616  105000                                  CLRB    R0              ;MAKE THE NEXT TWO PAGES NON-RESIDENT
060620  010037  172224                          MOV     R0,SDPDR2       ;SUPERVISOR D-SPACE PAGE 2
060624  010037  172226                          MOV     R0,SDPDR3       ;SUPERVISOR D-SPACE PAGE 3
060630  012737  000000  172240                  MOV     #000,SIPAR0     ;MAP SUPER I-SPACE PAGE 0 TO PHY 0
060636  012737  000200  172242                  MOV     #200,SIPAR1     ;MAP SUPER I-SPACE PAGE 1 TO 4K - 8K
060644  012737  000400  172244                  MOV     #400,SIPAR2     ;MAP SUPER I-SPACE PAGE 2 TO 8K - 12K
060652  012737  000600  172246                  MOV     #600,SIPAR3     ;MAP SUPER I-SPACE PAGE 3 TO 12K - 16K
060660  012737  001000  172250                  MOV     #1000,SIPAR4    ;MAP SUPER I-SPACE PAGE 4 TO 16K - 20K
060666  012737  177600  172256                  MOV     #177600,SIPAR7  ;MAP SUPER I-SPACE PAGE 7 TO I/O PAGE
060674  012737  000000  172260                  MOV     #000,SDPAR0     ;MAP SUPER D-SPACE PAGE 0 TO PHY 0
060702  012737  000200  172262                  MOV     #200,SDPAR1     ;MAP SUPER D-SPACE PAGE 1 TO 4K - 8K
060710  012737  177600  172276                  MOV     #177600,SDPAR7  ;MAP SUPER D-SPACE PAGE 7 TO I/O PAGE
060716  012704  172516                          MOV     #MMR3,R4        ;PUT ADDRESS OF MMR3 IN R4
060722  012705  000002                          MOV     #BIT1,R5        ;PUT ENABLE SUPERVISOR D-SPACE BIT IN R5
060726  052737  040000  177776                  BIS     #40000,PSW      ;GO INTO SUPERVISOR MODE HERE
060734  105037  172210                          CLRB    SIPDR4          ;MAKE I-SPACE PAGE 4 NON-RESIDENT
060740  012737  060746  001110                  MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
                                        ;;
                                        ;;      THIS SECTION CHECKS SIGNAL SPACEA FOR DISABLING D-SPACE
                                        ;;
060746  050514              10$:        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
060750  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
060752  000400                          BR      1$              ;THIS INST FETCH SHOULD USE ROM OUT09
060754  040514              1$:         BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
060756  012737  060764  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
060764  050514              11$:        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
060766  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
060770  017777  017004  017002          MOV     @100000,@100000 ;THIS INST USES ROM OUT09
060776  040514                          BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
061000  012737  061006  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
061006  050514              12$:        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
061010  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
061012  005037  100000                  CLR     @#100000        ;THIS INST USES ROM OUT15 & DSTF7
061016  040514                          BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
                                        ;;
                                        ;;      THIS SECTION CHECKS SIGNAL SPACEB FOR DISABLING D-SPACE
                                        ;;
061020  012737  061026  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
061026  050514              13$:        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
```

```
7862  061030  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7863  061032  013700  100000                MOV     @#100000,R0     ;THIS INST USES ROM OUT13 & SRCF7
7864  061036  040514                        BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
7865  061040  012737  061046  001110        MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
7866  061046  050514              14$:      BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
7867  061050  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7868  061052  005727  000000                TST     #0              ;THIS INST USES ROM OUT14 & DSTF7
7869  061056  040514                        BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
7870                                ;;
7871                                ;;      THE FOLLOWING SECTION VERIFIES THAT I-SPACE IS NOT
7872                                ;;      FORCED DURING THE ADDRESSING CYCLES OF THE INSTRUCTIONS.
7873                                ;;
7874  061060  012737  061066  001110        MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
7875  061066  112737  000006  172210  16$:  MOVB    #006,SIPDR4     ;MAKE I-SPACE PAGE 4 RESIDENT
7876  061074  012700  100000                MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
7877  061100  012737  100000  100000        MOV     #100000,@#100000 ;LOAD NO. 100000 INTO ADDRESS 100000
7878  061106  105037  172210                CLRB    SIPDR4          ;MAKE I-SPACE PAGE 4 NON-RESIDENT
7879  061112  050514                        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
7880  061114  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7881  061116  012730  100000                MOV     #100000,@(R0)+  ;THIS INST USES ROM OUT15 & -DSTF7
7882  061122  040514                        BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
7883  061124  012737  061132  001110        MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
7884  061132  012700  100000              17$:MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
7885  061136  050514                        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
7886  061140  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7887  061142  011001                        MOV     (R0),R1         ;THIS INST USES ROM OUT13 & -SRCF7
7888  061144  040514                        BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
7889  061146  012737  061154  001110        MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
7890  061154  012700  100000              18$:MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
7891  061160  050514                        BIS     R5,(R4)         ;ENABLE SUPER D-SPACE
7892  061162  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7893  061164  005720                        TST     (R0)+           ;THIS INST USES ROM OUT14 & -DSTF7
7894  061166  040514                        BIC     R5,(R4)         ;DISABLE SUPER D-SPACE
7895  061170  042737  040000  177776        BIC     #40000,PSW      ;GO BACK TO KERNEL MODE
7896  061176  012737  060542  001110        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
7897
7898
7899                                ;;*****************************************************************
7900                                ;*TEST 75        PROPER ENABLING OF USER D-SPACE
7901                                ;*
7902                                ;*      THIS TEST SHOWS THE PROPER FUNCTIONING OF USER D-SPACE.
7903                                ;*      BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
7904                                ;*      DURING THIS TEST FORCING I-SPACE.
7905                                ;*
7906                                ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
7907                                ;*      ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'.  THIS SUBROUTINE
7908                                ;*      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
7909                                ;*
7910                                ;;*****************************************************************
7911  061204                        TST75:
7912  061204  000004                        SCOPE
7913  061206  012737  061656  001314        MOV     #TST76,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
7914                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
7915  061214  012700  077406        20$:    MOV     #77406,R0       ;MAKE THE NEXT FEW PAGES 4K, R/W, UP
```

```
7916  061220  010037  177600                  MOV     R0,UIPDR0       ;USER I-SPACE PAGE 0
7917  061224  010037  177602                  MOV     R0,UIPDR1       ;USER I-SPACE PAGE 1
7918  061230  010037  177604                  MOV     R0,UIPDR2       ;USER I-SPACE PAGE 2
7919  061234  010037  177606                  MOV     R0,UIPDR3       ;USER I-SPACE PAGE 3
7920  061240  010037  177610                  MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
7921  061244  010037  177616                  MOV     R0,UIPDR7       ;USER I-SPACE PAGE 7
7922  061250  010037  177620                  MOV     R0,UDPDR0       ;USER D-SPACE PAGE 0
7923  061254  010037  177622                  MOV     R0,UDPDR1       ;USER D-SPACE PAGE 1
7924  061260  010037  177630                  MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
7925  061264  010037  177636                  MOV     R0,UDPDR7       ;USER D-SPACE PAGE 7
7926  061270  105000                          CLRB    R0              ;MAKE THE NEXT TWO PAGES NON-RESIDENT
7927  061272  010037  177624                  MOV     R0,UDPDR2       ;USER D-SPACE PAGE 2
7928  061276  010037  177626                  MOV     R0,UDPDR3       ;USER D-SPACE PAGE 3
7929  061302  012737  000000  177640          MOV     #000,UIPAR0     ;MAP USER I-SPACE PAGE 0 TO PHY 0
7930  061310  012737  000200  177642          MOV     #200,UIPAR1     ;MAP USER I-SPACE PAGE 1 TO 4K - 8K
7931  061316  012737  000400  177644          MOV     #400,UIPAR2     ;MAP USER I-SPACE PAGE 2 TO 8K - 12K
7932  061324  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER I-SPACE PAGE 3 TO 12K - 16K
7933  061332  012737  001000  177650          MOV     #1000,UIPAR4    ;MAP USER I-SPACE PAGE 4 TO 16K - 20K
7934  061340  012737  177600  177656          MOV     #177600,UIPAR7  ;MAP USER I-SPACE PAGE 7 TO I/O PAGE
7935  061346  012737  000000  177660          MOV     #000,UDPAR0     ;MAP USER D-SPACE PAGE 0 TO PHY 0
7936  061354  012737  000200  177662          MOV     #200,UDPAR1     ;MAP USER D-SPACE PAGE 1 TO 4K - 8K
7937  061362  012737  177600  177676          MOV     #177600,UDPAR7  ;MAP USER D-SPACE PAGE 7 TO I/O PAGE
7938  061370  012704  172516                  MOV     #MMR3,R4        ;PUT ADDRESS OF MMR3 IN R4
7939  061374  012705  000001                  MOV     #BIT0,R5        ;PUT ENABLE USER D-SPACE BIT IN R5
7940  061400  052737  140000  177776          BIS     #140000,PSW     ;GO INTO USER MODE HERE
7941  061406  105037  177610                  CLRB    UIPDR4          ;MAKE I-SPACE PAGE 4 NON-RESIDENT
7942  061412  012737  061420  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
7943  061420  050514                  10$:    BIS     R5,(R4)         ;ENABLE USER D-SPACE
7944  061422  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7945  061424  000400                          BR      1$              ;THIS INST FETCH SHOULD USE ROM OUT09
7946  061426  040514                  1$:     BIC     R5,(R4)         ;DISABLE USER D-SPACE
7947  061430  012737  061436  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
7948  061436  050514                  11$:    BIS     R5,(R4)         ;ENABLE USER D-SPACE
7949  061440  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7950  061442  017777  016332  016330          MOV     @100000,@100000 ;THIS INST USES ROM OUT09
7951  061450  040514                          BIC     R5,(R4)         ;DISABLE USER D-SPACE
7952  061452  012737  061460  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
7953  061460  050514                  12$:    BIS     R5,(R4)         ;ENABLE USER D-SPACE
7954  061462  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7955  061464  005037  100000                  CLR     @#100000        ;THIS INST USES ROM OUT15 & DSTF7
7956  061470  040514                          BIC     R5,(R4)         ;DISABLE USER D-SPACE
7957                                  ;;
7958                                  ;;      THIS SECTION CHECKS SIGNAL SPACEA FOR DISABLING D-SPACE
7959                                  ;;
7960  061472  012737  061500  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
7961  061500  050514                  13$:    BIS     R5,(R4)         ;ENABLE USER D-SPACE
7962  061502  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7963  061504  013700  100000                  MOV     @#100000,R0     ;THIS INST USES ROM OUT13 & SRCF7
7964  061510  040514                          BIC     R5,(R4)         ;DISABLE USER D-SPACE
7965  061512  012737  061520  001110          MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
7966  061520  050514                  14$:    BIS     R5,(R4)         ;ENABLE USER D-SPACE
7967  061522  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
7968  061524  005727  000000                  TST     #0              ;THIS INST USES ROM OUT14 & DSTF7
7969  061530  040514                          BIC     R5,(R4)         ;DISABLE USER D-SPACE
7970                                  ;;
7971                                  ;;      THIS SECTION OF CODE CHECKS TO BE SURE THAT I-SPACE IS NOT
```

```
                                        ;;      FORCED DURING ADDRESSING CYCLES OF INSTRUCTIONS
                                        ;;
061532  012737  061540  001110            MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
061540  112737  000006  177610  16$:      MOVB    #006,UIPDR4     ;MAKE I-SPACE PAGE 4 RESIDENT
061546  012700  100000                    MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
061552  012737  100000  100000            MOV     #100000,@#100000 ;LOAD NO. 100000 INTO ADDRESS 100000
061560  105037  177610                    CLRB    UIPDR4          ;MAKE I-SPACE PAGE 4 NON-RESIDENT
061564  050514                            BIS     R5,(R4)         ;ENABLE USER D-SPACE
061566  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
061570  012730  100000                    MOV     #100000,@(R0)+  ;THIS INST USES ROM OUT15 & -DSTF7
061574  040514                            BIC     R5,(R4)         ;DISABLE USER D-SPACE
061576  012737  061604  001110            MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
061604  012700  100000          17$:      MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
061610  050514                            BIS     R5,(R4)         ;ENABLE USER D-SPACE
061612  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
061614  011001                            MOV     (R0),R1         ;THIS INST USES ROM OUT13 & -SRCF7
061616  040514                            BIC     R5,(R4)         ;DISABLE USER D-SPACE
061620  012737  061626  001110            MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
061626  012700  100000          18$:      MOV     #100000,R0      ;LOAD ADDRESS 100000 INTO R0
061632  050514                            BIS     R5,(R4)         ;ENABLE USER D-SPACE
061634  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
061636  005720                            TST     (R0)+           ;THIS INST USES ROM OUT14 & -DSTF7
061640  040514                            BIC     R5,(R4)         ;DISABLE USER D-SPACE
061642  042737  140000  177776            BIC     #140000,PSW     ;GO BACK TO KERNEL MODE
061650  012737  061214  001110            MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

                                        ;;**************************************************************
                                        ;*TEST 76        TRAPPING IN D-SPACE KERNEL MODE
                                        ;*
                                        ;*      THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM D-SPACE
                                        ;*      AND NOT I-SPACE.  THE I-SPACE VECTOR POINTS TO 10$, AND
                                        ;*      THE D-SPACE VECTOR POINTS TO 15$.  EACH PSW IN VIRTUAL 252 IS
                                        ;*      DIFFERENT SO THE PROGRAM CAN TELL WHICH AREA IT IS PICKED
                                        ;*      UP FROM.
                                        ;;**************************************************************
061656                          TST76:
061656  000004                            SCOPE
061660  012737  062124  001314            MOV     #TST77,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
061666  104416                            TBITO                   ;MAKE SURE T-BIT IS OFF FOR THIS TEST
061670  012737  061726  001110  20$:      MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
061676  012737  062000  000250            MOV     #10$,@#250      ;SET M.M.VEC TO HOLD BAD VECTOR
061704  012737  000000  000252            MOV     #000,@#252      ;PROC. STAT. IN 252 HAS PRIORITY OF ZERO
061712  012737  062006  000350            MOV     #15$,@#350      ;D-SPACE M.M.VECTOR IS AT 350
061720  012737  000340  000352            MOV     #340,@#352      ;PRIORITY FOR D-SPACE VECTOR IS 7
061726  012706  001000          1$:       MOV     #1000,KSP       ;SET UP KERNEL VECTOR
061732  042737  177776  177572            BIC     #177776,MMR0    ;CLEAR ALL ERROR BITS IN MMR0
                                        ;       NOW SET UP FOR AN ABORT IN KERNEL MODE D-SPACE ENABLED
                                        ;
061740  012737  077402  172330            MOV     #77402,KDPDR4   ;KERNEL D-SPACE PAGE 4 IS READ ONLY
061746  012737  001000  172370            MOV     #1000,KDPAR4    ;MAP D-SPACE PAGE 4 TO 16K
061754  052737  000004  172516            BIS     #BIT2,MMR3      ;ENABLE KERNEL D-SPACE MAPPING
061762  012737  000001  172360            MOV     #1,KDPAR0       ;MAP KERNEL D PAGE 0 TO 000100
061770  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
061772  012737  177777  100000            MOV     #-1,@#100000    ;TRY TO WRITE TO PAGE 4
```

```
8028  062000  013700  177776          10$:    MOV     PSW,R0          ;SAVE PROC. STAT. FOR COMPARE
8029  062004  000402                  15$:    BR      16$             ;BRANCH TO D-SPACE READ CODE
8030  062006  013700  177776          15$:    MOV     PSW,R0          ;SAVE PROC. STAT FOR COMPARE
8031  062012  005037  172360          16$:    CLR     KDPAR0          ;RE-MAP KERNEL D PAGE 0 TO PHYSICAL 0
8032  062016  012706  001100                  MOV     #KERSTK,KSP     ;RE-SET STACK POINTER AFTER D-SPACE
8033                                                                  ;ABORT.
8034  062022  013737  177572  001246          MOV     MMR0,PMMR0      ;SAVE MMR0 FOR COMPARE
8035  062030  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR COMPARE
8036  062036  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR COMPARE
8037  062044  122700  000340                  CMPB    #340,R0         ;DID YOU PICK UP THE CORRECT PSW
8038  062050  001401                          BEQ     2$              ;BRANCH IF PSW IS 340
8039  062052  104130                          ERROR   130             ;WRONG PSW PICKED UP IN ABORT SEQUENCE
8040  062054  022737  020031  001246  2$:     CMP     #020031,PMMR0   ;EXPECTING READ ONLY ABORT
8041                                                                  ;KERNEL MODE, D-SPACE, PAGE 4
8042  062062  001401                          BEQ     3$              ;BRANCH IF CONDITION IS CORRECT
8043  062064  104131                          ERROR   131             ;WRONG M.M. ABORT CONDITION
8044  062066  042737  177776  177572  3$:     BIC     #177776,MMR0    ;CLEAR MMR0 FOR EXIT OF TEST
8045  062074  042737  000004  172516          BIC     #BIT2,MMR3      ;TURN OFF KERNEL D-SPACE ENABLE
8046  062102  012737  025054  000250          MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP VECTOR
8047  062110  012737  000340  000252          MOV     #340,MMVEC+2    ;PRIORITY EQUALS 7
8048  062116  012737  061670  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
8049
8050
8051                                  .SBTTL  ********** ENTRY POINT 7 --- STARTING ADDRESS 230 **********
8052                                  .SBTTL  ********** A & W BIT LOGIC TEST AND DUAL MAPPING TESTS *********
8053                                  ;*
8054                                  ;*      THIS GROUP OF TESTS CHECKS OUT THE A-BIT AND W-BIT LOGIC ON
8055                                  ;*      'SAPD' AND THEN USES THAT LOGIC TO VERIFY THAT THERE IS NO
8056                                  ;*      DUAL MAPPING OF PAR/PDR PAIRS BETWEEN GROUPS OR INSIDE A GROUP.
8057                                  ;*      ALL A-BITS AND W-BITS ARE SET TO ENSURE THAT THERE ISN'T A BAD
8058                                  ;*      CHIP IN THE PDR'S.
8059                                  ;*
8060
8061
8062                                  ;;*********************************************************************
8063                                  ;*TEST 77       TEST A-BIT AND W-BIT LOGIC
8064                                  ;*
8065                                  ;*      THIS TEST CHECKS ALL THE LOGIC FOR THE A-BIT AND W-BIT ON
8066                                  ;*      'SAPD'.  THE BITS ARE SET ONE AT A TIME THEN A REFERENCE
8067                                  ;*      TO THAT PAGE CAUSING AN ABORT IS MADE TO SEE IF THE BIT REMAINS
8068                                  ;*      SET.  LAST EITHER THE PAR OR PDR IS WRITTEN TO SEE THAT THE
8069                                  ;*      BITS ARE CLEARED DURING A PAR OR PDR LOAD.
8070                                  ;*
8071                                  ;;*********************************************************************
8072  062124                          TST77:
8073  062124  000004                          SCOPE
8074  062126  012737  062736  001314          MOV     #TST100,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
8075                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
8076  062134  104420                          TBITR                   ;RESTORE THE T-BIT TO ITS CONDITION
8077                                                                  ;BEFORE THE LAST TEST.
8078  062136                          ENTPT7:
8079  062136  012737  062276  001106          MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
8080  062144  012737  062276  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
8081  062152  012737  000077  001102          MOV     #77,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
8082  062160  013737  001102  177570          MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
8083  062166  012737  077406  172300          MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
```

```
8084  062174  012737  077406  172302          MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
8085  062202  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
8086  062210  012737  077406  172306          MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
8087  062216  012737  077406  172316          MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
8088  062224  012737  000000  172340          MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
8089  062232  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
8090  062240  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
8091  062246  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
8092  062254  012737  177600  172356          MOV     #177600,KIPAR7  ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
8093  062262  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
8094  062270  012737  000020  172516          MOV     #BIT4,MMR3      ;ENABLE 22-BIT RELOCATION IF NOT ON
8095                                          .EQUIV  BIT6,WBIT       ;LABEL 'WBIT' IS EQUIVALENT TO BIT 6
8096                                          .EQUIV  BIT7,ABIT       ;LABEL 'ABIT' IS EQUIVALENT TO BIT 7
8097  062276  012737  000006  172310  20$:    MOV     #00006,KIPDR4   ;PAGE 4 HAS A PAGE LENGTH OF 1
8098  062304  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP PAGE 4 TO 16K
8099  062312  012737  062726  000250          MOV     #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
8100  062320  012737  062326  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
8101  062326  000240                  11$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8102  062330  012737  012345  100000  1$:     MOV     #12345,@#100000 ;WRITE INTO PAGE 4
8103  062336  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK W-BIT
8104  062344  032737  000100  001170          BIT     #WBIT,$TMP0     ;SEE IF W-BIT IS SET
8105  062352  001001                          BNE     2$              ;BRANCH IF W-BIT IS SET
8106  062354  104102                          ERROR   102             ;W-BIT NOT SET
8107  062356  012737  062364  001110  2$:     MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
8108  062364  000240                  12$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8109  062366  012737  012345  100100          MOV     #12345,@#100100 ;WRITE INTO PAGE 4, BLOCK 2
8110  062374  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4TO SEE IF W-BIT REMAINED SET
8111  062402  032737  000100  001170          BIT     #WBIT,$TMP0     ;SEE IF W-BIT REMAINED SET
8112  062410  001001                          BNE     3$              ;BRANCH IF W-BIT IS SET
8113  062412  104103                          ERROR   103             ;W-BIT DID NOT REMAIN SET
8114  062414  012737  062422  001110  3$:     MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
8115  062422  000240                  13$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8116  062424  112737  000004  172310          MOVB    #004,KIPDR4     ;CHANGE PAGE 4'S ACCESS CONTROL FIELD
8117  062432  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK W-BIT CLEARING
8118  062440  022737  000004  001170          CMP     #00004,$TMP0    ;SEE IF W-BIT GOT CLEARED ON PDR WRITE
8119  062446  001401                          BEQ     4$              ;BRANCH IF W-BIT GOT CLEARED
8120  062450  104104                          ERROR   104             ;W-BIT DIDN'T GET CLEARED ON PDR WRITE
8121  062452  012737  062460  001110  4$:     MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
8122  062460  052737  001000  177572  14$:    BIS     #BIT9,MMR0      ;ENABLE M.M. TRAPS
8123  062466  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8124  062470  013700  100000                  MOV     @#100000,R0     ;READ FROM PAGE 4, BLOCK 1
8125  062474  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK A-BIT
8126  062502  032737  000200  001170          BIT     #ABIT,$TMP0     ;SEE IF A-BIT WAS SET DURING READ
8127  062510  001001                          BNE     5$              ;BRANCH IF A-BIT WAS SET
8128  062512  104105                          ERROR   105             ;A-BIT DIDN'T GET SET
8129  062514  012737  062522  001110  5$:     MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
8130  062522  052737  001000  177572  15$:    BIS     #BIT9,MMR0      ;ENABLE M.M. TRAPS
8131  062530  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8132  062532  013700  100100                  MOV     @#100100,R0     ;READ FROM PAGE 4, BLOCK 2
8133  062536  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO SEE IF A-BIT REMAINED SET
8134  062544  032737  000200  001170          BIT     #ABIT,$TMP0     ;SEE IF A-BIT IS STILL SET
8135  062552  001001                          BNE     6$              ;BRANCH IF A-BIT IS STILL SET
8136  062554  104106                          ERROR   106             ;A-BIT DIDN'T REMAIN SET
8137  062556  012737  062564  001110  6$:     MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
8138  062564  000240                  16$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8139  062566  012737  001000  172350          MOV     #1000,KIPAR4    ;WRITE PAR 4 TO SEE IF A-BIT CLEARS
```

```
8140  062574  013737  172310  001170          MOV     KIPDR4,$TMP0    ;READ PDR 4 TO CHECK A-BIT
8141  062602  022737  000004  001170          CMP     #00004,$TMP0    ;SEE IF A-BIT IS CLEAR
8142  062610  001401                          BEQ     7$              ;BRANCH IF A-BIT IS CLEAR
8143  062612  104107                          ERROR   107             ;A-BIT DIDN'T CLEAR ON PAR WRITE
8144  062614  012737  062622  001110  7$:     MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
8145  062622  000240                  17$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8146  062624  012737  123456  170200          MOV     #123456,MAPL0   ;LOAD A MAP REGISTER TO CHECK 'INT REG'
8147  062632  013737  172316  001170          MOV     KIPDR7,$TMP0    ;READ PDR 7 TO CHECK W-BIT
8148  062640  032737  000100  001170          BIT     #WBIT,$TMP0     ;SEE IF W-BIT SET ON MAP REGISTER WRITE
8149  062646  001001                          BNE     8$              ;BRANCH IF W-BIT WAS SET
8150  062650  104110                          ERROR   110             ;W-BIT DID NOT GET SET
8151  062652  012737  062660  001110  8$:     MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
8152  062660  000240                  18$:    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8153  062662  012737  077406  172310          MOV     #77406,KIPDR4   ;LOAD AN INTERNAL REGISTER, SHOULDN'T
8154                                                                  ;AFFECT PAGE 7'S W-BIT
8155  062670  013737  172316  001170          MOV     KIPDR7,$TMP0    ;READ PDR 7 TO CHECK W-BIT AGAIN
8156  062676  032737  000100  001170          BIT     #WBIT,$TMP0     ;SEE IF W-BIT IS STILL SET
8157  062704  001001                          BNE     9$              ;BRANCH IF W-BIT IS STILL SET
8158  062706  104111                          ERROR   111             ;W-BIT NOT SET AFTER WRITING PDR 4
8159  062710  012737  062276  001110  9$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
8160  062716  012737  025054  000250          MOV     #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP ROUTINE
8161  062724  000404                          BR      TST100          ;;BRANCH TO NEXT TEST
8162
8163  062726  042737  177776  177572  10$:    BIC     #177776,MMR0    ;CLEAR MMR0 FOR NEXT READ OR WRITE
8164  062734  000002                          RTI                     ;RETURN TO TEST AND CHECK A OR W BIT
8165
8166
8167                                          ;*
8168                                          ;*      THESE NEXT SIX (6) TESTS USE A.C.F. = 5 (TRAP ON WRITE) TO
8169                                          ;*      TEST ALL A & W BITS.  THE TESTS ALSO CHECK FOR DUAL MAPPING
8170                                          ;*      AMONG GROUPS OR INSIDE A GROUP OF PAR/PDR'S.  THIS IS DONE BY
8171                                          ;*      WRITING ONLY ONE PAGE IN A MODE OF OPERATION THEN CHECKING ALL
8172                                          ;*      PDR'S TO SEE THAT ONLY THE ONE UNDER TEST HAS BOTH ITS A & W
8173                                          ;*      BITS SET.  THIS TEST IS RUN IN ALL MODES, WITH D-SPACE DISABLED
8174                                          ;*      AND THEN AGAIN WITH D-SPACE ENABLED, SO THAT ALL THE REGISTER
8175                                          ;*      PAIRS ARE TESTED.
8176                                          ;*
8177                                          ;;*****************************************************************
8178                                          ;*TEST 100      DUAL MAPPING KERNEL MODE I-SPACE
8179                                          ;*
8180                                          ;*      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8181                                          ;*      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
8182                                          ;*      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
8183                                          ;*      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8184                                          ;*      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
8185                                          ;*      THE KERNEL I-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
8186                                          ;*      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8187                                          ;*      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8188                                          ;*      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
8189                                          ;*      ADDRESS OF 'MAPL00' (17770200) WHICH SHOULD ALWAYS EXIST.
8190                                          ;*
8191                                          ;;*****************************************************************
8192  062736                                  TST100:
8193  062736  000004                                  SCOPE
8194  062740  012737  064162  001314                  MOV     #TST101,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
8195                                                                          ;TEST FOR ESCAPE ON PARITY ERRORS
```

```
                                        ;       THE FOLLOWING  CODE IS USED TO INITIALIZE THE PAR'S FOR
                                        ;       THE NEXT TESTS, SO THAT THE CODE WILL RUN WITH MEMORY
                                        ;       MANAGEMENT FULLY ENABLED.
                                        ;
062746  012737  000000  172340          MOV     #0,KIPAR0       ;
062754  012737  000200  172342          MOV     #200,KIPAR1     ;
062762  012737  000400  172344          MOV     #400,KIPAR2     ;
062770  012737  000600  172346          MOV     #600,KIPAR3     ;
062776  012737  001000  172350          MOV     #1000,KIPAR4    ;
063004  012737  001000  172352          MOV     #1000,KIPAR5    ;
063012  012737  001000  172354          MOV     #1000,KIPAR6    ;
063020  012737  177600  172356          MOV     #177600,KIPAR7  ;
063026  012737  000000  172360          MOV     #0,KDPAR0       ;
063034  012737  000200  172362          MOV     #200,KDPAR1     ;
063042  012737  000400  172364          MOV     #400,KDPAR2     ;
063050  012737  000600  172366          MOV     #600,KDPAR3     ;
063056  012737  001000  172370          MOV     #1000,KDPAR4    ;
063064  012737  001000  172372          MOV     #1000,KDPAR5    ;
063072  012737  001000  172374          MOV     #1000,KDPAR6    ;
063100  012737  177600  172376          MOV     #177600,KDPAR7  ;
063106  012737  000000  172240          MOV     #0,SIPAR0       ;
063114  012737  000200  172242          MOV     #200,SIPAR1     ;
063122  012737  000400  172244          MOV     #400,SIPAR2     ;
063130  012737  000600  172246          MOV     #600,SIPAR3     ;
063136  012737  001000  172250          MOV     #1000,SIPAR4    ;
063144  012737  001000  172252          MOV     #1000,SIPAR5    ;
063152  012737  001000  172254          MOV     #1000,SIPAR6    ;
063160  012737  177600  172256          MOV     #177600,SIPAR7  ;
063166  012737  000000  172260          MOV     #0,SDPAR0       ;
063174  012737  000200  172262          MOV     #200,SDPAR1     ;
063202  012737  000400  172264          MOV     #400,SDPAR2     ;
063210  012737  000600  172266          MOV     #600,SDPAR3     ;
063216  012737  001000  172270          MOV     #1000,SDPAR4    ;
063224  012737  001000  172272          MOV     #1000,SDPAR5    ;
063232  012737  001000  172274          MOV     #1000,SDPAR6    ;
063240  012737  177600  172276          MOV     #177600,SDPAR7  ;
063246  012737  000000  177640          MOV     #0,UIPAR0       ;
063254  012737  000200  177642          MOV     #200,UIPAR1     ;
063262  012737  000400  177644          MOV     #400,UIPAR2     ;
063270  012737  000600  177646          MOV     #600,UIPAR3     ;
063276  012737  001000  177650          MOV     #1000,UIPAR4    ;
063304  012737  001000  177652          MOV     #1000,UIPAR5    ;
063312  012737  001000  177654          MOV     #1000,UIPAR6    ;
063320  012737  177600  177656          MOV     #177600,UIPAR7  ;
063326  012737  000000  177660          MOV     #0,UDPAR0       ;
063334  012737  000200  177662          MOV     #200,UDPAR1     ;
063342  012737  000400  177664          MOV     #400,UDPAR2     ;
063350  012737  000600  177666          MOV     #600,UDPAR3     ;
063356  012737  001000  177670          MOV     #1000,UDPAR4    ;
063364  012737  001000  177672          MOV     #1000,UDPAR5    ;
063372  012737  001000  177674          MOV     #1000,UDPAR6    ;
063400  012737  177600  177676          MOV     #177600,UDPAR7  ;
063406  012737  063424  001106          MOV     #20$,$LPADR     ;SET LOOP ON TEST POINTER TO 20$
063414  012737  063424  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
063422  104416                          TBITO                   ;MAKE SURE THAT T-BIT IS OFF FOR
                                                                ;THE SIX DUAL MAPPING TESTS
```

```
 8252  063424  012737  063574  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
 8253  063432  012737  000000  177776           MOV     #00000,PSW      ;GO TO KERNEL MODE
 8254  063440  012703  172300                   MOV     #KIPDR0,R3      ;LOAD FIRST ADDRESS OF KERNEL PDR'S
 8255                                                                   ;THAT WILL BE TESTED IN THIS TEST
 8256  063444  012704  000010                   MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
 8257  063450  012705  010200                   MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
 8258  063454  012700  077405           19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
 8259  063460  005037  001170                   CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
 8260  063464  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8261  063470  012701  177620                   MOV     #UDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8262  063474  010021                   1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8263  063476  077202                           SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
 8264  063500  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8265  063504  012701  177600                   MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8266  063510  010021                   2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8267  063512  077202                           SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
 8268  063514  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8269  063520  012701  172220                   MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8270  063524  010021                   3$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8271  063526  077202                           SOB     R2,3$           ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
 8272  063530  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8273  063534  012701  172200                   MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8274  063540  010021                   4$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8275  063542  077202                           SOB     R2,4$           ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
 8276  063544  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8277  063550  012701  172320                   MOV     #KDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8278  063554  010021                   5$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8279  063556  077202                           SOB     R2,5$           ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
 8280  063560  012702  000010                   MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
 8281  063564  012701  172300                   MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
 8282  063570  010021                   6$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 8283  063572  077202                           SOB     R2,6$           ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
 8284  063574  012700  077405           21$:    MOV     #77405,R0       ;MUST RE-INIT THESE PDRS IF ERROR
 8285  063600  010037  172300                   MOV     R0,KIPDR0       ;LOAD KERNEL PDR0
 8286  063604  010037  172302                   MOV     R0,KIPDR1       ;LOAD KERNEL PDR1
 8287  063610  010037  172316                   MOV     R0,KIPDR7       ;LOAD KERNEL PDR7
 8288  063614  010037  172300                   MOV     R0,KIPDR0       ;LOAD PDR0 OF PRESENT SPACE
 8289  063620  010037  172316                   MOV     R0,KIPDR7       ;LOAD PDR7 OF PRESENT SPACE
 8290  063624  000240                           NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 8291  063626  011515                           MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
 8292  063630  012702  000020                   MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
 8293  063634  012701  172300                   MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
 8294  063640  011100           7$:     MOV     (R1),R0         ;READ PDR INTO R0
 8295  063642  022700  077705                   CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
 8296  063646  001023                           BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
 8297  063650  020103                           CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
 8298  063652  001417                           BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
 8299  063654  104112                           ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
 8300  063656  012700  077405                   MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8301  063662  010037  172300                   MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
 8302  063666  010037  172302                   MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
 8303  063672  010037  172316                   MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
 8304  063676  010037  172300                   MOV     R0,KIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8305  063702  010037  172316                   MOV     R0,KIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8306  063706  011515                           MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8307                                                                   ;WERE TESTING PAGE SEVEN
```

```
 8308  063710  000402                    BR      9$              ;GO UPDATE R1 FOR NEXT READ
 8309  063712  005237  001170    8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8310  063716  062701  000002    9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8311  063722  077232                    SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
 8312  063724  012702  000020            MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
 8313  063730  012701  172200            MOV     #SIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
 8314  063734  011100            10$:    MOV     (R1),R0         ;READ PDR INTO R0
 8315  063736  022700  077705            CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
 8316  063742  001023                    BNE     12$             ;BRANCH IF THIS IS NOT THE ONE
 8317  063744  020103                    CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
 8318  063746  001417                    BEQ     11$             ;BRANCH IF ADDRESS IS CORRECT
 8319  063750  104112                    ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
 8320  063752  012700  077405            MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8321  063756  010037  172300            MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
 8322  063762  010037  172302            MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
 8323  063766  010037  172316            MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
 8324  063772  010037  172300            MOV     R0,KIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8325  063776  010037  172316            MOV     R0,KIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8326  064002  011515                    MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8327                                                            ;WERE TESTING PAGE SEVEN
 8328  064004  000402                    BR      12$             ;GO UPDATE R1 FOR NEXT READ
 8329  064006  005237  001170    11$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8330  064012  062701  000002    12$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8331  064016  077232                    SOB     R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
 8332  064020  012702  000020            MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
 8333  064024  012701  177600            MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
 8334  064030  011100            13$:    MOV     (R1),R0         ;READ PDR INTO R0
 8335  064032  022700  077705            CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
 8336  064036  001023                    BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
 8337  064040  020103                    CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
 8338  064042  001417                    BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
 8339  064044  104112                    ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
 8340  064046  012700  077405            MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8341  064052  010037  172300            MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
 8342  064056  010037  172302            MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
 8343  064062  010037  172316            MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
 8344  064066  010037  172300            MOV     R0,KIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8345  064072  010037  172316            MOV     R0,KIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8346  064076  011515                    MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8347                                                            ;WERE TESTING PAGE SEVEN
 8348  064100  000402                    BR      15$             ;GO UPDATE R1 FOR NEXT READ
 8349  064102  005237  001170    14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8350  064106  062701  000002    15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8351  064112  077232                    SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
 8352  064114  005737  001170            TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
 8353  064120  001002                    BNE     16$             ;BRANCH IF THERE WAS
 8354  064122  011300                    MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
 8355  064124  104113                    ERROR   113             ;NO PDR ADDRESSES MATCHED
 8356  064126  062703  000002    16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
 8357  064132  062705  020000            ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
 8358  064136  005304                    DEC     R4              ;DECREMENT COUNTER
 8359  064140  001402                    BEQ     17$             ;BRANCH IF COUNTER IS ZERO
 8360  064142  000137  063454            JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
 8361  064146                    17$:
 8362  064146  012737  000340  177776    MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
 8363  064154  012737  063424  001110    MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                        ;;*****************************************************************
                        ;*TEST 101      DUAL MAPPING SUPERVISOR MODE I-SPACE
                        ;;
                        ;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
                        ;;      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
                        ;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
                        ;;      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
                        ;;      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
                        ;;      THE SUPERVISOR I-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
                        ;;      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
                        ;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
                        ;;      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
                        ;;      ADDRESS OF 'MAPL00'' (17770200) WHICH SHOULD ALWAYS EXIST.
                        ;;
                        ;;*****************************************************************
064162                  TST101:
                                SCOPE
064162  000004
064164  012737  064730  001314          MOV     #TST102,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
064172  012737  064342  001110  20$:     MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
064200  012737  040000  177776          MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
064206  012703  172200                  MOV     #SIPDR0,R3      ;LOAD FIRST ADDRESS OF SUPERVISOR PDR'S
                                                                ;THAT WILL BE TESTED IN THIS TEST
064212  012704  000010                  MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
064216  012705  010200                  MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
064222  012700  077405          19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
064226  005037  001170                  CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
064232  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064236  012701  172320                  MOV     #KDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064242  010021                  1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064244  077202                          SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
064246  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064252  012701  172300                  MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064256  010021                  2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064260  077202                          SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
064262  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064266  012701  177620                  MOV     #UDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064272  010021                  3$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064274  077202                          SOB     R2,3$           ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
064276  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064302  012701  177600                  MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064306  010021                  4$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064310  077202                          SOB     R2,4$           ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
064312  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064316  012701  172220                  MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064322  010021                  5$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064324  077202                          SOB     R2,5$           ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
064326  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
064332  012701  172200                  MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
064336  010021                  6$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
064340  077202                          SOB     R2,6$           ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
064342  012700  077405          21$:    MOV     #77405,R0       ;MUST RE-INIT THESE PDRS IF ERROR
064346  010037  172300                  MOV     R0,KIPDR0       ;LOAD KERNEL PDR0
064352  010037  172302                  MOV     R0,KIPDR1       ;LOAD KERNEL PDR1
```

```
8420  064356  010037  172316          MOV     R0,KIPDR7       ;LOAD KERNEL PDR7
8421  064362  010037  172200          MOV     R0,SIPDR0       ;LOAD PDR0 OF PRESENT SPACE
8422  064366  010037  172216          MOV     R0,SIPDR7       ;LOAD PDR7 OF PRESENT SPACE
8423  064372  000240                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8424  064374  011515                  MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
8425  064376  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8426  064402  012701  172300          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8427  064406  011100          7$:     MOV     (R1),R0         ;READ PDR INTO R0
8428  064410  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8429  064414  001023                  BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
8430  064416  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8431  064420  001417                  BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
8432  064422  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8433  064424  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8434  064430  010037  172300          MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
8435  064434  010037  172302          MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
8436  064440  010037  172316          MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
8437  064444  010037  172200          MOV     R0,SIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8438  064450  010037  172216          MOV     R0,SIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8439  064454  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8440                                                          ;WERE TESTING PAGE SEVEN
8441  064456  000402                  BR      9$              ;GO UPDATE R1 FOR NEXT READ
8442  064460  005237  001170  8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8443  064464  062701  000002  9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8444  064470  077232                  SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
8445  064472  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8446  064476  012701  172200          MOV     #SIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8447  064502  011100          10$:    MOV     (R1),R0         ;READ PDR INTO R0
8448  064504  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8449  064510  001023                  BNE     12$             ;BRANCH IF THIS IS NOT THE ONE
8450  064512  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8451  064514  001417                  BEQ     11$             ;BRANCH IF ADDRESS IS CORRECT
8452  064516  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8453  064520  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8454  064524  010037  172300          MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
8455  064530  010037  172302          MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
8456  064534  010037  172316          MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
8457  064540  010037  172200          MOV     R0,SIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8458  064544  010037  172216          MOV     R0,SIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8459  064550  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8460                                                          ;WERE TESTING PAGE SEVEN
8461  064552  000402                  BR      12$             ;GO UPDATE R1 FOR NEXT READ
8462  064554  005237  001170  11$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8463  064560  062701  000002  12$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8464  064564  077232                  SOB     R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
8465  064566  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8466  064572  012701  177600          MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8467  064576  011100          13$:    MOV     (R1),R0         ;READ PDR INTO R0
8468  064600  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8469  064604  001023                  BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
8470  064606  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8471  064610  001417                  BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
8472  064612  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8473  064614  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8474  064620  010037  172300          MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
8475  064624  010037  172302          MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
```

```
8476  064630  010037  172316          MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
8477  064634  010037  172200          MOV     R0,SIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8478  064640  010037  172216          MOV     R0,SIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8479  064644  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8480                                                          ;WERE TESTING PAGE SEVEN
8481  064646  000402                  BR      15$             ;GO UPDATE R1 FOR NEXT READ
8482  064650  005237  001170  14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8483  064654  062701  000002  15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8484  064660  077232                  SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
8485  064662  005737  001170          TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
8486  064666  001002                  BNE     16$             ;BRANCH IF THERE WAS
8487  064670  011300                  MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
8488  064672  104113                  ERROR   113             ;NO PDR ADDRESSES MATCHED
8489  064674  062703  000002  16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
8490  064700  062705  020000          ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
8491  064704  005304                  DEC     R4              ;DECREMENT COUNTER
8492  064706  001402                  BEQ     17$             ;BRANCH IF COUNTER IS ZERO
8493  064710  000137  064222          JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
8494  064714                  17$:
8495  064714  012737  000340  177776  MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
8496  064722  012737  064172  001110  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
8497
8498
8499                          ;;*****************************************************************
8500                          ;*TEST 102      DUAL MAPPING USER MODE I-SPACE
8501                          ;;
8502                          ;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8503                          ;;      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
8504                          ;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
8505                          ;;      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8506                          ;;      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
8507                          ;;      THE USER I-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
8508                          ;;      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8509                          ;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8510                          ;;      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
8511                          ;;      ADDRESS OF 'MAPL00'' (17770200) WHICH SHOULD ALWAYS EXIST.
8512                          ;;
8513                          ;;*****************************************************************
8514  064730                  TST102:
8515  064730  000004                  SCOPE
8516  064732  012737  065476  001314  MOV     #TST103,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
8517                                                          ;TEST FOR ESCAPE ON PARITY ERRORS
8518  064740  012737  065110  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
8519  064746  012737  140000  177776  MOV     #140000,PSW     ;GO TO USER MODE
8520  064754  012703  177600          MOV     #UIPDR0,R3      ;LOAD FIRST ADDRESS OF USER PDR'S
8521                                                          ;THAT WILL BE TESTED IN THIS TEST
8522  064760  012704  000010          MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
8523  064764  012705  010200          MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
8524  064770  012700  077405  19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
8525  064774  005037  001170          CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
8526  065000  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8527  065004  012701  172220          MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8528  065010  010021          1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8529  065012  077202                  SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
8530  065014  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8531  065020  012701  172200          MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
```

```
8532  065024  010021                2$:   MOV   R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
8533  065026  077202                      SOB   R2,2$          ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
8534  065030  012702  000010              MOV   #10,R2         ;SET COUNT TO LOAD 8 ADDRESSES
8535  065034  012701  172320              MOV   #KDPDR0,R1     ;PUT ADDRESS OF FIRST PDR IN R1
8536  065040  010021                3$:   MOV   R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
8537  065042  077202                      SOB   R2,3$          ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
8538  065044  012702  000010              MOV   #10,R2         ;SET COUNT TO LOAD 8 ADDRESSES
8539  065050  012701  172300              MOV   #KIPDR0,R1     ;PUT ADDRESS OF FIRST PDR IN R1
8540  065054  010021                4$:   MOV   R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
8541  065056  077202                      SOB   R2,4$          ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
8542  065060  012702  000010              MOV   #10,R2         ;SET COUNT TO LOAD 8 ADDRESSES
8543  065064  012701  177620              MOV   #UDPDR0,R1     ;PUT ADDRESS OF FIRST PDR IN R1
8544  065070  010021                5$:   MOV   R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
8545  065072  077202                      SOB   R2,5$          ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
8546  065074  012702  000010              MOV   #10,R2         ;SET COUNT TO LOAD 8 ADDRESSES
8547  065100  012701  177600              MOV   #UIPDR0,R1     ;PUT ADDRESS OF FIRST PDR IN R1
8548  065104  010021                6$:   MOV   R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
8549  065106  077202                      SOB   R2,6$          ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
8550  065110  012700  077405        21$:  MOV   #77405,R0      ;MUST RE-INIT THESE PDRS IF ERROR
8551  065114  010037  172300              MOV   R0,KIPDR0      ;LOAD KERNEL PDR0
8552  065120  010037  172302              MOV   R0,KIPDR1      ;LOAD KERNEL PDR1
8553  065124  010037  172316              MOV   R0,KIPDR7      ;LOAD KERNEL PDR7
8554  065130  010037  177600              MOV   R0,UIPDR0      ;LOAD PDR0 OF PRESENT SPACE
8555  065134  010037  177616              MOV   R0,UIPDR7      ;LOAD PDR7 OF PRESENT SPACE
8556  065140  000240                      NOP                  ;THIS IS A SYNC POINT FOR SCOPING
8557  065142  011515                      MOV   (R5),(R5)      ;WRITE INTO PAGE UNDER TEST
8558  065144  012702  000020              MOV   #20,R2         ;SET COUNTER TO READ NEXT 20 REGISTERS
8559  065150  012701  172300              MOV   #KIPDR0,R1     ;LOAD ADDRESS OF BEGINNING PDR
8560  065154  011100                7$:   MOV   (R1),R0        ;READ PDR INTO R0
8561  065156  022700  077705              CMP   #77705,R0      ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8562  065162  001023                      BNE   9$             ;BRANCH IF THIS IS NOT THE ONE
8563  065164  020103                      CMP   R1,R3          ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8564  065166  001417                      BEQ   8$             ;BRANCH IF ADDRESS IS CORRECT
8565  065170  104112                      ERROR 112            ;A & W BITS GOT SET IN WRONG PDR
8566  065172  012700  077405              MOV   #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
8567  065176  010037  172300              MOV   R0,KIPDR0      ;RELOAD KERNEL PDR0
8568  065202  010037  172302              MOV   R0,KIPDR1      ;RELOAD KERNEL PDR1
8569  065206  010037  172316              MOV   R0,KIPDR7      ;RELOAD KERNEL PDR7
8570  065212  010037  177600              MOV   R0,UIPDR0      ;RELOAD PAGE 0 OF PRESENT SPACE
8571  065216  010037  177616              MOV   R0,UIPDR7      ;RE-LOAD I/O PAGE PDR IF ERROR
8572  065222  011515                      MOV   (R5),(R5)      ;TRY WRITE AGAIN, IN CASE YOU
8573                                                           ;WERE TESTING PAGE SEVEN
8574  065224  000402                      BR    9$             ;GO UPDATE R1 FOR NEXT READ
8575  065226  005237  001170        8$:   INC   $TMP0          ;SET FLAG SINCE ADDRESSES MATCHED
8576  065232  062701  000002        9$:   ADD   #2,R1          ;POINT TO NEXT PDR TO BE READ
8577  065236  077232                      SOB   R2,7$          ;BRANCH TO 7$ IF ALL PDR'S NOT READ
8578  065240  012702  000020              MOV   #20,R2         ;SET COUNTER TO READ NEXT 20 REGISTERS
8579  065244  012701  172200              MOV   #SIPDR0,R1     ;LOAD ADDRESS OF BEGINNING PDR
8580  065250  011100                10$:  MOV   (R1),R0        ;READ PDR INTO R0
8581  065252  022700  077705              CMP   #77705,R0      ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8582  065256  001023                      BNE   12$            ;BRANCH IF THIS IS NOT THE ONE
8583  065260  020103                      CMP   R1,R3          ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8584  065262  001417                      BEQ   11$            ;BRANCH IF ADDRESS IS CORRECT
8585  065264  104112                      ERROR 112            ;A & W BITS GOT SET IN WRONG PDR
8586  065266  012700  077405              MOV   #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
8587  065272  010037  172300              MOV   R0,KIPDR0      ;RELOAD KERNEL PDR0
```

```
8588  065276  010037  172302              MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
8589  065302  010037  172316              MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
8590  065306  010037  177600              MOV     R0,UIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8591  065312  010037  177616              MOV     R0,UIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8592  065316  011515                      MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8593                                                               ;WERE TESTING PAGE SEVEN
8594  065320  000402                      BR      12$             ;GO UPDATE R1 FOR NEXT READ
8595  065322  005237  001170      11$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8596  065326  062701  000002      12$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8597  065332  077232                      SOB     R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
8598  065334  012702  000020              MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8599  065340  012701  177600              MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8600  065344  011100      13$:    MOV     (R1),R0         ;READ PDR INTO R0
8601  065346  022700  077705              CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8602  065352  001023                      BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
8603  065354  020103                      CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8604  065356  001417                      BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
8605  065360  104112                      ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8606  065362  012700  077405              MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8607  065366  010037  172300              MOV     R0,KIPDR0       ;RELOAD KERNEL PDR0
8608  065372  010037  172302              MOV     R0,KIPDR1       ;RELOAD KERNEL PDR1
8609  065376  010037  172316              MOV     R0,KIPDR7       ;RELOAD KERNEL PDR7
8610  065402  010037  177600              MOV     R0,UIPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8611  065406  010037  177616              MOV     R0,UIPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8612  065412  011515                      MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8613                                                               ;WERE TESTING PAGE SEVEN
8614  065414  000402                      BR      15$             ;GO UPDATE R1 FOR NEXT READ
8615  065416  005237  001170      14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8616  065422  062701  000002      15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8617  065426  077232                      SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
8618  065430  005737  001170              TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
8619  065434  001002                      BNE     16$             ;BRANCH IF THERE WAS
8620  065436  011300                      MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
8621  065440  104113                      ERROR   113             ;NO PDR ADDRESSES MATCHED
8622  065442  062703  000002      16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
8623  065446  062705  020000              ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
8624  065452  005304                      DEC     R4              ;DECREMENT COUNTER
8625  065454  001402                      BEQ     17$             ;BRANCH IF COUNTER IS ZERO
8626  065456  000137  064770              JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
8627  065462                      17$:
8628  065462  012737  000340  177776      MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
8629  065470  012737  064740  001110      MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
8630
8631
8632                              ;;*****************************************************************
8633                              ;*TEST 103       DUAL MAPPING KERNEL MODE D-SPACE
8634                              ;;
8635                              ;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8636                              ;;      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
8637                              ;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
8638                              ;;      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8639                              ;;      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
8640                              ;;      THE KERNEL D-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
8641                              ;;      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8642                              ;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8643                              ;;      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
```

```
                                        ;;      ADDRESS OF 'MAPL00'' (17770200) WHICH SHOULD ALWAYS EXIST.
                                        ;;
                                        ;;*****************************************************************
065476                                  TST103:
065476  000004                                  SCOPE
065500  012737  066260  001314                  MOV     #TST104,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                        ;TEST FOR ESCAPE ON PARITY ERRORS
065506  012737  065664  001110          20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
065514  012737  000000  177776                  MOV     #00000,PSW      ;GO TO KERNEL MODE
065522  052737  000007  172516                  BIS     #7,MMR3         ;ENABLE ALL D-SPACE MAPPING
065530  012703  172320                          MOV     #KDPDR0,R3      ;LOAD FIRST ADDRESS OF KERNEL PDR'S
                                                                        ;THAT WILL BE TESTED IN THIS TEST
065534  012704  000010                          MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
065540  012705  010200                          MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
065544  012700  077405                  19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
065550  005037  001170                          CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
065554  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065560  012701  177600                          MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065564  010021                          1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065566  077202                                  SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
065570  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065574  012701  177620                          MOV     #UDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065600  010021                          2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065602  077202                                  SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
065604  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065610  012701  172200                          MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065614  010021                          3$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065616  077202                                  SOB     R2,3$           ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
065620  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065624  012701  172220                          MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065630  010021                          4$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065632  077202                                  SOB     R2,4$           ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
065634  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065640  012701  172300                          MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065644  010021                          5$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065646  077202                                  SOB     R2,5$           ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
065650  012702  000010                          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
065654  012701  172320                          MOV     #KDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
065660  010021                          6$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
065662  077202                                  SOB     R2,6$           ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
065664  012700  077405                  21$:    MOV     #77405,R0       ;MUST RE-INIT THESE PDRS IF ERROR
065670  010037  172320                          MOV     R0,KDPDR0       ;LOAD KERNEL PDR0
065674  010037  172322                          MOV     R0,KDPDR1       ;LOAD KERNEL PDR1
065700  010037  172336                          MOV     R0,KDPDR7       ;LOAD KERNEL PDR7
065704  010037  172320                          MOV     R0,KDPDR0       ;LOAD PDR0 OF PRESENT SPACE
065710  010037  172336                          MOV     R0,KDPDR7       ;LOAD PDR7 OF PRESENT SPACE
065714  000240                                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
065716  011515                                  MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
065720  012702  000020                          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
065724  012701  172300                          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
065730  011100                          7$:     MOV     (R1),R0         ;READ PDR INTO R0
065732  022700  077705                          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
065736  001023                                  BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
065740  020103                                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
065742  001417                                  BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
065744  104112                                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
```

```
 8700  065746  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8701  065752  010037  172320          MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
 8702  065756  010037  172322          MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
 8703  065762  010037  172336          MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
 8704  065766  010037  172320          MOV     R0,KDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8705  065772  010037  172336          MOV     R0,KDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8706  065776  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8707                                                          ;WERE TESTING PAGE SEVEN
 8708  066000  000402                  BR      9$              ;GO UPDATE R1 FOR NEXT READ
 8709  066002  005237  001170  8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8710  066006  062701  000002  9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8711  066012  077232                  SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
 8712  066014  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
 8713  066020  012701  172200          MOV     #SIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
 8714  066024  011100          10$:    MOV     (R1),R0         ;READ PDR INTO R0
 8715  066026  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
 8716  066032  001023                  BNE     12$             ;BRANCH IF THIS IS NOT THE ONE
 8717  066034  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
 8718  066036  001417                  BEQ     11$             ;BRANCH IF ADDRESS IS CORRECT
 8719  066040  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
 8720  066042  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8721  066046  010037  172320          MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
 8722  066052  010037  172322          MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
 8723  066056  010037  172336          MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
 8724  066062  010037  172320          MOV     R0,KDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8725  066066  010037  172336          MOV     R0,KDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8726  066072  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8727                                                          ;WERE TESTING PAGE SEVEN
 8728  066074  000402                  BR      12$             ;GO UPDATE R1 FOR NEXT READ
 8729  066076  005237  001170  11$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8730  066102  062701  000002  12$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8731  066106  077232                  SOB     R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
 8732  066110  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
 8733  066114  012701  177600          MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
 8734  066120  011100          13$:    MOV     (R1),R0         ;READ PDR INTO R0
 8735  066122  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
 8736  066126  001023                  BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
 8737  066130  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
 8738  066132  001417                  BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
 8739  066134  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
 8740  066136  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
 8741  066142  010037  172320          MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
 8742  066146  010037  172322          MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
 8743  066152  010037  172336          MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
 8744  066156  010037  172320          MOV     R0,KDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
 8745  066162  010037  172336          MOV     R0,KDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
 8746  066166  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
 8747                                                          ;WERE TESTING PAGE SEVEN
 8748  066170  000402                  BR      15$             ;GO UPDATE R1 FOR NEXT READ
 8749  066172  005237  001170  14$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
 8750  066176  062701  000002  15$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
 8751  066202  077232                  SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
 8752  066204  005737  001170          TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
 8753  066210  001002                  BNE     16$             ;BRANCH IF THERE WAS
 8754  066212  011300                  MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
 8755  066214  104113                  ERROR   113             ;NO PDR ADDRESSES MATCHED
```

```
                                16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
                                        ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
                                        DEC     R4              ;DECREMENT COUNTER
                                        BEQ     17$             ;BRANCH IF COUNTER IS ZERO
                                        JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
                                17$:
                                        BIC     #7,MMR3         ;DISABLE ALL D-SPACE MAPPING
                                        MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
                                        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST

;;*****************************************************************
;*TEST 104       DUAL MAPPING SUPERVISOR MODE D-SPACE
;;
;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
;;      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
;;      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
;;      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
;;      THE SUPERVISOR D-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
;;      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
;;      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
;;      ADDRESS OF 'MAPL00'' (17770200) WHICH SHOULD ALWAYS EXIST.
;;
;;*****************************************************************
TST104:
                                        SCOPE
                                        MOV     #TST105,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
                                        MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
                                        BIS     #7,MMR3         ;ENABLE ALL D-SPACE MAPPING
                                        MOV     #SDPDR0,R3      ;LOAD FIRST ADDRESS OF SUPERVISOR PDR'S
                                                                ;THAT WILL BE TESTED IN THIS TEST
                                        MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
                                        MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
                                19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
                                        CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
                                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
                                        MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
                                1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
                                        SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
                                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
                                        MOV     #KDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
                                2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
                                        SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
                                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
                                        MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
                                3$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
                                        SOB     R2,3$           ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
                                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
                                        MOV     #UDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
                                4$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
                                        SOB     R2,4$           ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
                                        MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
```

```
8812  066422  012701  172200          MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8813  066426  010021                  5$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8814  066430  077202                          SOB     R2,5$           ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
8815  066432  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8816  066436  012701  172220                  MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8817  066442  010021                  6$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8818  066444  077202                          SOB     R2,6$           ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
8819  066446  012700  077405          21$:    MOV     #77405,R0       ;MUST RE-INIT THESE PDRS IF ERROR
8820  066452  010037  172320                  MOV     R0,KDPDR0       ;LOAD KERNEL PDR0
8821  066456  010037  172322                  MOV     R0,KDPDR1       ;LOAD KERNEL PDR1
8822  066462  010037  172336                  MOV     R0,KDPDR7       ;LOAD KERNEL PDR7
8823  066466  010037  172220                  MOV     R0,SDPDR0       ;LOAD PDR0 OF PRESENT SPACE
8824  066472  010037  172236                  MOV     R0,SDPDR7       ;LOAD PDR7 OF PRESENT SPACE
8825  066476  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8826  066500  011515                          MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
8827  066502  012702  000020                  MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8828  066506  012701  172300                  MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8829  066512  011100                  7$:     MOV     (R1),R0         ;READ PDR INTO R0
8830  066514  022700  077705                  CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8831  066520  001023                          BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
8832  066522  020103                          CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8833  066524  001417                          BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
8834  066526  104112                          ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8835  066530  012700  077405                  MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8836  066534  010037  172320                  MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
8837  066540  010037  172322                  MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
8838  066544  010037  172336                  MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
8839  066550  010037  172220                  MOV     R0,SDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8840  066554  010037  172236                  MOV     R0,SDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8841  066560  011515                          MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8842                                                                  ;WERE TESTING PAGE SEVEN
8843  066562  000402                          BR      9$              ;GO UPDATE R1 FOR NEXT READ
8844  066564  005237  001170          8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8845  066570  062701  000002          9$:     ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8846  066574  077232                          SOB     R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
8847  066576  012702  000020                  MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8848  066602  012701  172200                  MOV     #SIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8849  066606  011100                  10$:    MOV     (R1),R0         ;READ PDR INTO R0
8850  066610  022700  077705                  CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8851  066614  001023                          BNE     12$             ;BRANCH IF THIS IS NOT THE ONE
8852  066616  020103                          CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8853  066620  001417                          BEQ     11$             ;BRANCH IF ADDRESS IS CORRECT
8854  066622  104112                          ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8855  066624  012700  077405                  MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8856  066630  010037  172320                  MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
8857  066634  010037  172322                  MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
8858  066640  010037  172336                  MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
8859  066644  010037  172220                  MOV     R0,SDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8860  066650  010037  172236                  MOV     R0,SDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8861  066654  011515                          MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8862                                                                  ;WERE TESTING PAGE SEVEN
8863  066656  000402                          BR      12$             ;GO UPDATE R1 FOR NEXT READ
8864  066660  005237  001170          11$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8865  066664  062701  000002          12$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8866  066670  077232                          SOB     R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
8867  066672  012702  000020                  MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
```

```
8868  066676  012701  177600          MOV     #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8869  066702  011100          13$:    MOV     (R1),R0         ;READ PDR INTO R0
8870  066704  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8871  066710  001023                  BNE     15$             ;BRANCH IF THIS IS NOT THE ONE
8872  066712  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8873  066714  001417                  BEQ     14$             ;BRANCH IF ADDRESS IS CORRECT
8874  066716  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8875  066720  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8876  066724  010037  172320          MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
8877  066730  010037  172322          MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
8878  066734  010037  172336          MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
8879  066740  010037  172220          MOV     R0,SDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8880  066744  010037  172236          MOV     R0,SDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8881  066750  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8882                                                          ;WERE TESTING PAGE SEVEN
8883  066752  000402                  BR      15$             ;GO UPDATE R1 FOR NEXT READ
8884  066754  005237  001170  14$:    INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
8885  066760  062701  000002  15$:    ADD     #2,R1           ;POINT TO NEXT PDR TO BE READ
8886  066764  077232                  SOB     R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
8887  066766  005737  001170          TST     $TMP0           ;SEE IF THERE WAS A CORRECT PDR
8888  066772  001002                  BNE     16$             ;BRANCH IF THERE WAS
8889  066774  011300                  MOV     (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
8890  066776  104113                  ERROR   113             ;NO PDR ADDRESSES MATCHED
8891  067000  062703  000002  16$:    ADD     #2,R3           ;POINT TO NEXT PDR UNDER TEST
8892  067004  062705  020000          ADD     #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
8893  067010  005304                  DEC     R4              ;DECREMENT COUNTER
8894  067012  001402                  BEQ     17$             ;BRANCH IF COUNTER IS ZERO
8895  067014  000137  066326          JMP     19$             ;JUMP TO LOAD PDR'S AGAIN
8896  067020                  17$:
8897  067020  042737  000007  172516  BIC     #7,MMR3         ;DISABLE ALL D-SPACE MAPPING
8898  067026  012737  000340  177776  MOV     #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
8899  067034  012737  066270  001110  MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
8900
8901
8902                          ;;**************************************************************
8903                          ;*TEST 105       DUAL MAPPING USER MODE D-SPACE
8904                          ;;
8905                          ;;      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8906                          ;;      (4K PAGE,TRAP ON WRITE).  THEN THE VIRTUAL ADDRESS IS SET TO
8907                          ;;      010200 AND THAT WORD IS WRITTEN INTO ITSELF.  THIS WRITE WILL
8908                          ;;      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8909                          ;;      SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
8910                          ;;      THE USER D-SPACE P.D.R. UNDER TEST.  NOW ALL OF THE
8911                          ;;      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8912                          ;;      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8913                          ;;      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
8914                          ;;      ADDRESS OF 'MAPL00' (17770200) WHICH SHOULD ALWAYS EXIST.
8915                          ;;
8916                          ;;**************************************************************
8917  067042                  TST105:
8918  067042  000004                  SCOPE
8919  067044  012737  067624  001314  MOV     #TST106,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
8920                                                          ;TEST FOR ESCAPE ON PARITY ERRORS
8921  067052  012737  067230  001110  20$:    MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
8922  067060  012737  140000  177776  MOV     #140000,PSW     ;GO TO USER MODE
8923  067066  052737  000007  172516  BIS     #7,MMR3         ;ENABLE ALL D-SPACE MAPPING
```

```
8924  067074  012703  177620          MOV     #UDPDR0,R3      ;LOAD FIRST ADDRESS OF USER PDR'S
8925                                                          ;THAT WILL BE TESTED IN THIS TEST
8926  067100  012704  000010          MOV     #10,R4          ;TEST THE NEXT EIGHT PDR'S
8927  067104  012705  010200          MOV     #10200,R5       ;LOAD STARTING VIRTUAL ADDRESS INTO R5
8928  067110  012700  077405  19$:    MOV     #77405,R0       ;ALL PAGES WILL BE TRAP ON WRITE
8929  067114  005037  001170          CLR     $TMP0           ;CLEAR CORRECT PDR SET INDICATOR
8930  067120  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8931  067124  012701  172200          MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8932  067130  010021          1$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8933  067132  077202                  SOB     R2,1$           ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
8934  067134  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8935  067140  012701  172220          MOV     #SDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8936  067144  010021          2$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8937  067146  077202                  SOB     R2,2$           ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
8938  067150  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8939  067154  012701  172300          MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8940  067160  010021          3$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8941  067162  077202                  SOB     R2,3$           ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
8942  067164  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8943  067170  012701  172320          MOV     #KDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8944  067174  010021          4$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8945  067176  077202                  SOB     R2,4$           ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
8946  067200  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8947  067204  012701  177600          MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8948  067210  010021          5$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8949  067212  077202                  SOB     R2,5$           ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
8950  067214  012702  000010          MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
8951  067220  012701  177620          MOV     #UDPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
8952  067224  010021          6$:     MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
8953  067226  077202                  SOB     R2,6$           ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
8954  067230  012700  077405  21$:    MOV     #77405,R0       ;MUST RE-INIT THESE PDRS IF ERROR
8955  067234  010037  172320          MOV     R0,KDPDR0       ;LOAD KERNEL PDR0
8956  067240  010037  172322          MOV     R0,KDPDR1       ;LOAD KERNEL PDR1
8957  067244  010037  172336          MOV     R0,KDPDR7       ;LOAD KERNEL PDR7
8958  067250  010037  177620          MOV     R0,UDPDR0       ;LOAD PDR0 OF PRESENT SPACE
8959  067254  010037  177636          MOV     R0,UDPDR7       ;LOAD PDR7 OF PRESENT SPACE
8960  067260  000240                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
8961  067262  011515                  MOV     (R5),(R5)       ;WRITE INTO PAGE UNDER TEST
8962  067264  012702  000020          MOV     #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8963  067270  012701  172300          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8964  067274  011100          7$:     MOV     (R1),R0         ;READ PDR INTO R0
8965  067276  022700  077705          CMP     #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8966  067302  001023                  BNE     9$              ;BRANCH IF THIS IS NOT THE ONE
8967  067304  020103                  CMP     R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8968  067306  001417                  BEQ     8$              ;BRANCH IF ADDRESS IS CORRECT
8969  067310  104112                  ERROR   112             ;A & W BITS GOT SET IN WRONG PDR
8970  067312  012700  077405          MOV     #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8971  067316  010037  172320          MOV     R0,KDPDR0       ;RELOAD KERNEL PDR0
8972  067322  010037  172322          MOV     R0,KDPDR1       ;RELOAD KERNEL PDR1
8973  067326  010037  172336          MOV     R0,KDPDR7       ;RELOAD KERNEL PDR7
8974  067332  010037  177620          MOV     R0,UDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8975  067336  010037  177636          MOV     R0,UDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8976  067342  011515                  MOV     (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8977                                                          ;WERE TESTING PAGE SEVEN
8978  067344  000402                  BR      9$              ;GO UPDATE R1 FOR NEXT READ
8979  067346  005237  001170  8$:     INC     $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
```

```
8980  067352  062701  000002          9$:   ADD   #2,R1          ;POINT TO NEXT PDR TO BE READ
8981  067356  077232                        SOB   R2,7$           ;BRANCH TO 7$ IF ALL PDR'S NOT READ
8982  067360  012702  000020                MOV   #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
8983  067364  012701  172200                MOV   #SIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
8984  067370  011100                  10$:  MOV   (R1),R0         ;READ PDR INTO R0
8985  067372  022700  077705                CMP   #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8986  067376  001023                        BNE   12$             ;BRANCH IF THIS IS NOT THE ONE
8987  067400  020103                        CMP   R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8988  067402  001417                        BEQ   11$             ;BRANCH IF ADDRESS IS CORRECT
8989  067404  104112                        ERROR 112             ;A & W BITS GOT SET IN WRONG PDR
8990  067406  012700  077405                MOV   #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
8991  067412  010037  172320                MOV   R0,KDPDR0       ;RELOAD KERNEL PDR0
8992  067416  010037  172322                MOV   R0,KDPDR1       ;RELOAD KERNEL PDR1
8993  067422  010037  172336                MOV   R0,KDPDR7       ;RELOAD KERNEL PDR7
8994  067426  010037  177620                MOV   R0,UDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
8995  067432  010037  177636                MOV   R0,UDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
8996  067436  011515                        MOV   (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
8997                                                              ;WERE TESTING PAGE SEVEN
8998  067440  000402                        BR    12$             ;GO UPDATE R1 FOR NEXT READ
8999  067442  005237  001170          11$:  INC   $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
9000  067446  062701  000002          12$:  ADD   #2,R1           ;POINT TO NEXT PDR TO BE READ
9001  067452  077232                        SOB   R2,10$          ;BRANCH TO 10$ IF ALL PDR'S NOT READ
9002  067454  012702  000020                MOV   #20,R2          ;SET COUNTER TO READ NEXT 20 REGISTERS
9003  067460  012701  177600                MOV   #UIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
9004  067464  011100                  13$:  MOV   (R1),R0         ;READ PDR INTO R0
9005  067466  022700  077705                CMP   #77705,R0       ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9006  067472  001023                        BNE   15$             ;BRANCH IF THIS IS NOT THE ONE
9007  067474  020103                        CMP   R1,R3           ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9008  067476  001417                        BEQ   14$             ;BRANCH IF ADDRESS IS CORRECT
9009  067500  104112                        ERROR 112             ;A & W BITS GOT SET IN WRONG PDR
9010  067502  012700  077405                MOV   #77405,R0       ;RE-SET PAGES MODIFIED BY ERROR
9011  067506  010037  172320                MOV   R0,KDPDR0       ;RELOAD KERNEL PDR0
9012  067512  010037  172322                MOV   R0,KDPDR1       ;RELOAD KERNEL PDR1
9013  067516  010037  172336                MOV   R0,KDPDR7       ;RELOAD KERNEL PDR7
9014  067522  010037  177620                MOV   R0,UDPDR0       ;RELOAD PAGE 0 OF PRESENT SPACE
9015  067526  010037  177636                MOV   R0,UDPDR7       ;RE-LOAD I/O PAGE PDR IF ERROR
9016  067532  011515                        MOV   (R5),(R5)       ;TRY WRITE AGAIN, IN CASE YOU
9017                                                              ;WERE TESTING PAGE SEVEN
9018  067534  000402                        BR    15$             ;GO UPDATE R1 FOR NEXT READ
9019  067536  005237  001170          14$:  INC   $TMP0           ;SET FLAG SINCE ADDRESSES MATCHED
9020  067542  062701  000002          15$:  ADD   #2,R1           ;POINT TO NEXT PDR TO BE READ
9021  067546  077232                        SOB   R2,13$          ;BRANCH TO 13$ IF ALL PDR'S NOT READ
9022  067550  005737  001170                TST   $TMP0           ;SEE IF THERE WAS A CORRECT PDR
9023  067554  001002                        BNE   16$             ;BRANCH IF THERE WAS
9024  067556  011300                        MOV   (R3),R0         ;SAVE CONTENTS OF PDR UNDER TEST
9025  067560  104113                        ERROR 113             ;NO PDR ADDRESSES MATCHED
9026  067562  062703  000002          16$:  ADD   #2,R3           ;POINT TO NEXT PDR UNDER TEST
9027  067566  062705  020000                ADD   #20000,R5       ;CHANGE PAGE NUMBER IN VIRT. ADDR.
9028  067572  005304                        DEC   R4              ;DECREMENT COUNTER
9029  067574  001402                        BEQ   17$             ;BRANCH IF COUNTER IS ZERO
9030  067576  000137  067110                JMP   19$             ;JUMP TO LOAD PDR'S AGAIN
9031  067602                          17$:
9032  067602  042737  000007  172516        BIC   #7,MMR3         ;DISABLE ALL D-SPACE MAPPING
9033  067610  012737  000340  177776        MOV   #340,PSW        ;RETURN TO KERNEL MODE, PRIORITY 7
9034  067616  012737  067052  001110        MOV   #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
9035
```

```
                                .SBTTL  ************ ENTRY POINT 8 --- STARTING ADDRESS 234 ************
                                .SBTTL  ***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****
                                ;*
                                ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
                                ;*      THE 'MOVE FROM PREVIOUS'' AND MOVE TO PREVIOUS'' INSTRUCTIONS.
                                ;*      THE LOGIC IS PRIMARILY ON ''SSRB'', THE 'ROM OUTXX'' SIGNALS ARE
                                ;*      GENERATED BY THE ROMS ON ''SSRA''.
                                ;*


                                ;;*********************************************************************
                                ;*TEST 106      MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE
                                ;*
                                ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05'.
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                ;*      THEIR ADDRESSES).
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
                                ;;*********************************************************************
067624                          TST106:
067624  000004                          SCOPE
067626  012737  070426  001314        MOV     #TST107,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
067634  104420                        TBITR                   ;RESTORE T-BIT TO ITS STATUS BEFORE
                                                                ;THE SIX DUAL MAPPING TESTS
067636                          ENTPT8:
067636  012737  070040  001106        MOV     #20$,$LPADR     ;SET LOOP ADDRESS POINTER TO 20$
067644  012737  070040  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
067652  012737  000106  001102        MOV     #106,$TSTNM     ;LOAD TEST NUMBER INTO MEMORY
067660  013737  001102  177570        MOV     $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
067666  012737  077406  172300        MOV     #77406,KIPDR0   ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
067674  012737  077406  172302        MOV     #77406,KIPDR1   ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
067702  012737  077406  172304        MOV     #77406,KIPDR2   ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
067710  012737  077406  172306        MOV     #77406,KIPDR3   ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
067716  012737  077406  172316        MOV     #77406,KIPDR7   ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
067724  012737  000000  172340        MOV     #000,KIPAR0     ;MAP KERNEL I PAGE 0 TO 0 - 4K
067732  012737  000200  172342        MOV     #200,KIPAR1     ;MAP KERNEL I PAGE 1 TO 4K - 8K
067740  012737  000400  172344        MOV     #400,KIPAR2     ;MAP KERNEL I PAGE 2 TO 8K - 12K
067746  012737  000600  172346        MOV     #600,KIPAR3     ;MAP KERNEL I PAGE 3 TO 12K - 16K
067754  012737  177600  172356        MOV     #177600,KIPAR7  ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
067762  012737  000001  177572        MOV     #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
067770  012737  000020  172516        MOV     #BIT4,MMR3      ;ENABLE 22-BIT RELOCATION IF NOT ON
067776  042737  000007  172516        BIC     #7,MMR3         ;MAKE SURE THAT ALL D-SPACE IS DISABLED
070004  012700  077406                MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
                                                                ;READ/WRITE, LENGTH 200 BLOCKS
070010  012702  000010                MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
070014  012701  172300                MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
070020  010021                19$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
070022  077202                        SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
070024  012737  070040  001106        MOV     #20$,$LPADR     ;SET LOOP POINTER TO 20$
070032  012737  070040  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
070040  012700  077400        20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
```

```
                                                        ;AND KERNEL I NON-RESIDENT
070044  010037  172330          MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
070050  010037  172230          MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
070054  010037  177630          MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
070060  010037  177610          MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
070064  012737  077406  172310  MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
070072  012737  077406  172210  MOV     #77406,SIPDR4   ;SUPER I-SPACE PAGE 4 READ/WRITE
070100  012737  001000  172350  MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
070106  012737  001000  172250  MOV     #1000,SIPAR4    ;MAP SUPER I PAGE 4 TO 16K
070114  012700  036514          MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
070120  010037  100000          MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 100000
070124  012737  070400  000250  MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
070132  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
070136  012737  070144  001110  MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
070144  012737  010340  177776  11$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
070152  000240                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
070154  006506                  1$:     MFPI    SSP             ;PUT SUPERVISOR STACK POINTER ON KERNEL
                                                        ;STACK
070156  022706  001100          CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 1$
070162  001407                  BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
070164  012601                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
070166  012702  000700          MOV     #SUPSTK,R2      ;EXPECTING TO GET 700 AS SSP
070172  020201                  CMP     R2,R1           ;DID YOU GET THE RIGHT POINTER?
070174  001403                  BEQ     2$              ;BRANCH IF YOU DID
070176  104114                  ERROR   114             ;WRONG THING WAS PUSHED ON STACK
070200  000401                  BR      2$              ;BRANCH TO NEXT TRY
070202  104115                  3$:     ERROR   115             ;NOTHING PUSHED ON STACK
070204                          2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE
                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
                                        ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                        ;INTO THE F/F'S ON SSRB.
                                        ;   *  D12.00    (001)
                                        ;   *  D12.10    (175)
                                        ;   *  MFP.00    (066)
                                        ;      MFP.10    (250)
                                        ;      SVC.80    (222)
                                        ;      SVC.90    (300)
                                        ;      FET.00    (217)
070204  012737  070212  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
070212  012737  010340  177776  12$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
070220  012702  100000          MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
070224  000240                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
070226  006512                  MFPI    (R2)            ;READ FROM PHYSICAL 100000
070230  012601                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
070232  020001                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
070234  001401                  BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
070236  104116                  ERROR   116             ;WRONG DATA WAS FETCHED
070240                          4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
                                        ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                        ;INTO THE F/F'S ON SSRB.
                                        ;   *  D12.01    (002)
                                        ;   *  D12.10    (175)
                                        ;   *  MFP.00    (066)
                                        ;      MFP.10    (250)
                                        ;      SVC.80    (222)
```

```
9148                                                ;       SVC.90    (300)
9149                                                ;       FET.00    (217)
9150  070240  012737  070246  001110                MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
9151  070246  012737  010340  177776  14$:          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
9152  070254  012702  100000                       MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
9153  070260  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
9154  070262  006522                               MFPI    (R2)+           ;READ FROM PHYSICAL 100000
9155  070264  012601                               MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
9156  070266  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
9157  070270  001401                               BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
9158  070272  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
9159  070274                             5$:       ;THE FOLLOWING WILL TEST DSTM=3 MFPI.  BELOW ARE THE
9160                                               ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9161                                               ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9162                                               ;INTO THE F/F'S ON SSRB.
9163                                               ;       D30.00    (003)
9164                                               ;       D30.10    (221)
9165                                               ;       D10.20    (233)
9166                                               ;   *   D10.50    (311)
9167                                               ;   *   D10.60    (177)
9168                                               ;   *   MFP.00    (066)
9169                                               ;       MFP.10    (250)
9170                                               ;       SVC.80    (222)
9171                                               ;       SVC.90    (300)
9172                                               ;       FET.00    (217)
9173  070274  012737  070302  001110                MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
9174  070302  012737  010340  177776  15$:          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
9175  070310  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
9176  070312  006537  100000                       MFPI    @#100000        ;READ FROM PHYSICAL 100000
9177  070316  012601                               MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
9178  070320  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
9179  070322  001401                               BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
9180  070324  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
9181  070326                             6$:       ;THE FOLLOWING WILL TEST DSTM=4 MFPI.  BELOW ARE THE
9182                                               ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9183                                               ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9184                                               ;INTO THE F/F'S ON SSRB.
9185                                               ;       D45.00    (004)
9186                                               ;   *   D10.30    (122)
9187                                               ;   *   D10.60    (177)
9188                                               ;   *   MFP.00    (066)
9189                                               ;       MFP.10    (250)
9190                                               ;       SVC.80    (222)
9191                                               ;       SVC.90    (300)
9192                                               ;       FET.00    (217)
9193  070326  012737  070334  001110                MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
9194  070334  012737  010340  177776  16$:          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
9195  070342  012702  100002                       MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
9196  070346  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
9197  070350  006542                               MFPI    -(R2)           ;READ FROM PHYSICAL 100000
9198  070352  012601                               MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
9199  070354  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
9200  070356  001401                               BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
9201  070360  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
9202  070362  012737  025054  000250  7$:          MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
9203  070370  012737  070040  001110                MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                                        BR      TST107          ;;BRANCH TO NEXT TEST

                                ;;**************************************************************
                                ;*TEST 107      MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
                                ;*
                                ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05'.
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                ;*      THEIR ADDRESSES).
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
                                ;;**************************************************************
                                TST107:
                                        SCOPE
                                        MOV     #TST110,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
                                                                ;AND KERNEL I NON-RESIDENT
                                        MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
                                        MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
                                        MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
                                        MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
                                        MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
                                        MOV     #77406,SIPDR4   ;SUPER I-SPACE PAGE 4 READ/WRITE
                                        MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
                                        MOV     #1000,SIPAR4    ;MAP SUPER I PAGE 4 TO 16K
                                        MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
                                1$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
                                        MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
                                        MTPI    SSP             ;LOAD SUPERVISOR STACK POINTER
                                        MFPI    SSP             ;READ SUPERVISOR STACK POINTER
                                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
                                        CMP     #7777,R1        ;WAS SUPERVISOR STACK POINTER CHANGED
                                        BEQ     2$              ;BRANCH IF IT WAS
                                        ERROR   120             ;SUPER STACK POINTER NOT CHANGED
                                2$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
                                        MOV     #SUPSTK,-(KSP)  ;GET READY TO RESTORE SUPERVISOR S. POINT
                                        MTPI    SSP             ;RESTORE SUPERVISOR STACK POINTER
                                3$:     ;THIS WILL TEST DSTM = 1 MTPI. BELOW ARE THE
                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                        ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                        ;INTO THE FLIP/FLOPS ON SSRB.
                                        ;   *  D12.80    (111)
                                        ;   *  D12.60    (155)
                                        ;      D12.20    (312)
                                        ;      FET.10    (260)
```

```
 9260  070564  012737  070602  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
 9261  070572  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9262  070576  012700  125252                  MOV     #125252,R0      ;LOAD TEST DATA INTO R0
 9263  070602  010046                  13$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
 9264  070604  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 9265  070610  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9266  070612  006612                          MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 100000
 9267  070614  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
 9268  070622  011201                          MOV     (R2),R1         ;READ FROM ADDRESS 100000
 9269  070624  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
 9270  070626  001401                          BEQ     4$              ;BRANCH IF STORE WAS CORRECT
 9271  070630  104121                          ERROR   121             ;INCORRECT STORE
 9272  070632                          4$:     ;THIS WILL TEST DSTM = 3 MTPI. BELOW ARE THE
 9273                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
 9274                                          ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9275                                          ;INTO THE FLIP/FLOPS ON SSRB.
 9276                                          ;       D30.80    113)
 9277                                          ;       D30.10    (221)
 9278                                          ;       D10.20    (233)
 9279                                          ;   *   D10.50    (311)
 9280                                          ;   *   D10.40    (157)
 9281                                          ;       FET.01    (331)
 9282  070632  012737  070652  001110          MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
 9283  070640  012737  010340  177776          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9284  070646  012700  052525                  MOV     #52525,R0       ;LOAD TEST DATA INTO R0
 9285  070652  010046                  14$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
 9286  070654  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 9287  070660  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9288  070662  006637  100000                  MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 100000
 9289  070666  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
 9290  070674  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 100000
 9291  070700  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
 9292  070702  001401                          BEQ     5$              ;BRANCH IF STORE WAS CORRECT
 9293  070704  104121                          ERROR   121             ;INCORRECT STORE
 9294  070706                          5$:     ;THIS WILL TEST DSTM = 4 MTPI. BELOW ARE THE
 9295                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
 9296                                          ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9297                                          ;INTO THE FLIP/FLOPS ON SSRB.
 9298                                          ;       D45.80    (115)
 9299                                          ;   *   D40.20    (121)
 9300                                          ;   *   D10.40    (157)
 9301                                          ;       FET.01    (331)
 9302  070706  012737  070726  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
 9303  070714  012737  010340  177776          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9304  070722  012700  125252                  MOV     #125252,R0      ;LOAD TEST DATA INTO R0
 9305  070726  010046                  15$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
 9306  070730  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9307  070734  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 9308  070740  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9309  070742  006642                          MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 100000
 9310  070744  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
 9311  070752  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 100000
 9312  070756  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
 9313  070760  001401                          BEQ     6$              ;BRANCH IF STORE WAS CORRECT
 9314  070762  104121                          ERROR   121             ;INCORRECT STORE
 9315  070764                          6$:     ;THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE
```

```
                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                        ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                        ;INTO THE FLIP/FLOPS ON SSRB.
                                        ;       D67.80    (117)
                                        ;       D67.00    (006)
                                        ;       D67.10    (251)
                                        ;   *   D10.30    (122)
                                        ;   *   D10.40    (157)
                                        ;       FET.01    (331)
070764  012737  071006  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
070772  012737  010340  177776          MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
071000  012700  052525                  MOV     #52525,R0       ;LOAD TEST DATA INTO R0
071004  005002                          CLR     R2              ;MAKE REGISTER 2 ZERO
071006  010046                  16$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
071010  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
071014  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
071016  006662  100000                  MTPI    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 100000
071022  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
071030  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 100000
071034  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
071036  001401                          BEQ     7$              ;BRANCH IF STORE WAS CORRECT
071040  104121                          ERROR   121             ;INCORRECT STORE
071042  012737  070436  001110  7$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
071050  012737  025054  000250          MOV     #MMTRAP,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
071056  000413                          BR      TST110          ;;BRANCH TO NEXT TEST

071060  013737  177572  001246  10$:    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
071066  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
071074  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
071102  104117                          ERROR   117             ;TRIED TO LOAD A N.R. PAGE 4
071104  000002                          RTI                     ;RETURN TO TEST


                                ;;******************************************************************
                                ;*TEST 110      MFPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
                                ;*
                                ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05', AND THAT
                                ;*      D-SPACE IS NOT ENABLED.  THIS IS DONE BY 'ROM OUT08 H' AND
                                ;*      'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                ;*      THEIR ADDRESSES).
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
                                ;;******************************************************************
071106                  TST110:
071106  000004                  SCOPE
071110  012737  071506  001314          MOV     #TST111,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
071116  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
                                                                ;READ/WRITE, LENGTH 200 BLOCKS
071122  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
071126  012701  172300                  MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
```

```
 9372  071132  010021                    19$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
 9373  071134  077202                            SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
 9374  071136  012737  071152  001106            MOV     #20$,$LPADR     ;SET LOOP POINTER TO 20$
 9375  071144  012737  071152  001110            MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
 9376  071152  012700  077400            20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
 9377                                                                    ;AND KERNEL I NON-RESIDENT
 9378  071156  010037  172330                    MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
 9379  071162  010037  172230                    MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
 9380  071166  010037  177630                    MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
 9381  071172  010037  177610                    MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
 9382  071176  012737  077406  172310            MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
 9383  071204  012737  077406  172210            MOV     #77406,SIPDR4   ;SUPER I-SPACE PAGE 4 READ/WRITE
 9384  071212  012737  001000  172350            MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
 9385  071220  012737  001000  172250            MOV     #1000,SIPAR4    ;MAP SUPER I PAGE 4 TO 16K
 9386  071226  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
 9387  071232  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 100000
 9388  071236  012737  071460  000250            MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
 9389  071244  052737  000002  172516            BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
 9390  071252  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
 9391                                            ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE
 9392                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9393                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9394                                            ;INTO THE F/F'S ON SSRB.
 9395                                            ;   *  D12.00    (001)
 9396                                            ;   *  D12.10    (175)
 9397                                            ;   *  MFP.00    (066)
 9398                                            ;      MFP.10    (250)
 9399                                            ;      SVC.80    (222)
 9400                                            ;      SVC.90    (300)
 9401                                            ;      FET.00    (217)
 9402  071256  012737  071264  001110            MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
 9403  071264  012737  010340  177776    12$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9404  071272  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9405  071276  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9406  071300  006512                            MFPI    (R2)            ;READ FROM PHYSICAL 100000
 9407  071302  012601                            MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 9408  071304  020001                            CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9409  071306  001401                            BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9410  071310  104116                            ERROR   116             ;WRONG DATA WAS FETCHED
 9411  071312                            4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
 9412                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9413                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9414                                            ;INTO THE F/F'S ON SSRB.
 9415                                            ;   *  D12.01    (002)
 9416                                            ;   *  D12.10    (175)
 9417                                            ;   *  MFP.00    (066)
 9418                                            ;      MFP.10    (250)
 9419                                            ;      SVC.80    (222)
 9420                                            ;      SVC.90    (300)
 9421                                            ;      FET.00    (217)
 9422  071312  012737  071320  001110            MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
 9423  071320  012737  010340  177776    14$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9424  071326  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9425  071332  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9426  071334  006522                            MFPI    (R2)+           ;READ FROM PHYSICAL 100000
 9427  071336  012601                            MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
```

```
 9428  071340  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9429  071342  001401                          BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9430  071344  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
 9431  071346                          5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.  BELOW ARE THE
 9432                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9433                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9434                                          ;INTO THE F/F'S ON SSRB.
 9435                                          ;       D30.00  (003)
 9436                                          ;       D30.10  (221)
 9437                                          ;       D10.20  (233)
 9438                                          ;   *   D10.50  (311)
 9439                                          ;   *   D10.60  (177)
 9440                                          ;   *   MFP.00  (066)
 9441                                          ;       MFP.10  (250)
 9442                                          ;       SVC.80  (222)
 9443                                          ;       SVC.90  (300)
 9444                                          ;       FET.00  (217)
 9445  071346  012737  071354  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
 9446  071354  012737  010340  177776  15$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9447  071362  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9448  071364  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 100000
 9449  071370  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 9450  071372  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9451  071374  001401                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9452  071376  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
 9453  071400                          6$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.  BELOW ARE THE
 9454                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9455                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9456                                          ;INTO THE F/F'S ON SSRB.
 9457                                          ;       D45.00  (004)
 9458                                          ;   *   D10.30  (122)
 9459                                          ;   *   D10.60  (177)
 9460                                          ;   *   MFP.00  (066)
 9461                                          ;       MFP.10  (250)
 9462                                          ;       SVC.80  (222)
 9463                                          ;       SVC.90  (300)
 9464                                          ;       FET.00  (217)
 9465  071400  012737  071406  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
 9466  071406  012737  010340  177776  16$:    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 9467  071414  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9468  071420  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9469  071422  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 100000
 9470  071424  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 9471  071426  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9472  071430  001401                          BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9473  071432  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
 9474  071434  012737  025054  000250  7$:     MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
 9475  071442  012737  071152  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 9476  071450  042737  000002  172516          BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
 9477  071456  000413                          BR      TST111          ;;BRANCH TO NEXT TEST
 9478
 9479
 9480  071460  013737  177572  001246  10$:    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
 9481  071466  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
 9482  071474  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
 9483  071502  104117                          ERROR   117             ;TRIED TO READ NON-RESIDENT PAGE
```

```
                                        RTI                     ;RETURN TO TEST


                                ;;*****************************************************************
                                ;*TEST 111       MTPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
                                ;*
                                ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05', AND THAT
                                ;*      D-SPACE IS NOT ENABLED.  THIS IS DONE BY 'ROM OUT08 H' AND
                                ;*      'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                ;*      THEIR ADDRESSES).
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
                                ;;*****************************************************************
071506                          TST111:
                                        SCOPE
071506  000004
071510  012737  072136  001314        MOV     #TST112,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
071516  012700  077400        20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
                                                                ;AND KERNEL I NON-RESIDENT
071522  010037  172330                MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
071526  010037  172230                MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
071532  010037  177630                MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
071536  010037  177610                MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
071542  012737  077406  172310        MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
071550  012737  077406  172210        MOV     #77406,SIPDR4   ;SUPER I-SPACE PAGE 4 READ/WRITE
071556  012737  001000  172350        MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
071564  012737  001000  172250        MOV     #1000,SIPAR4    ;MAP SUPER I PAGE 4 TO 16K
071572  012737  072110  000250        MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
071600  052737  000002  172516        BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
                                ;THIS WILL TEST DSTM = 1 MTPI. BELOW ARE THE
                                ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                ;INTO THE FLIP/FLOPS ON SSRB.
                                ;   *  D12.80    (111)
                                ;   *  D12.60    (155)
                                ;      D12.20    (312)
                                ;      FET.10    (260)
071606  012737  071624  001110        MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
071614  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
071620  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
071624  010046                13$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
071626  105037  172310                CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
071632  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
071634  006612                        MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 100000
071636  112737  000006  172310        MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
071644  011201                        MOV     (R2),R1         ;READ FROM ADDRESS 100000
071646  020001                        CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
071650  001401                        BEQ     4$              ;BRANCH IF STORE WAS CORRECT
071652  104121                        ERROR   121             ;INCORRECT STORE
071654                          4$:     ;THIS WILL TEST DSTM = 3 MTPI. BELOW ARE THE
```

```
                                                ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                                ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                                ;INTO THE FLIP/FLOPS ON SSRB.
                                                ;       D30.80    (113)
                                                ;       D30.10    (221)
                                                ;       D10.20    (233)
                                                ;   *   D10.50    (311)
                                                ;   *   D10.40    (157)
                                                ;       FET.01    (331)
071654  012737  071674  001110                    MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
071662  012737  010340  177776                    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
071670  012700  052525                            MOV     #52525,R0       ;LOAD TEST DATA INTO R0
071674  010046                        14$:        MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
071676  105037  172310                            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
071702  000240                                    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
071704  006637  100000                            MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 100000
071710  112737  000006  172310                    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
071716  013701  100000                            MOV     @#100000,R1     ;READ FROM ADDRESS 100000
071722  020001                                    CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
071724  001401                                    BEQ     5$              ;BRANCH IF STORE WAS CORRECT
071726  104121                                    ERROR   121             ;INCORRECT STORE
071730                                  5$:       ;THIS WILL TEST DSTM = 4 MTPI. BELOW ARE THE
                                                ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                                ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                                ;INTO THE FLIP/FLOPS ON SSRB.
                                                ;       D45.80    (115)
                                                ;   *   D40.20    (121)
                                                ;   *   D10.40    (157)
                                                ;       FET.01    (331)
071730  012737  071750  001110                    MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
071736  012737  010340  177776                    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
071744  012700  125252                            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
071750  010046                        15$:        MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
071752  012702  100002                            MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
071756  105037  172310                            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
071762  000240                                    NOP                     ;THIS IS A SYNC POINT FOR SCOPING
071764  006642                                    MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 100000
071766  112737  000006  172310                    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
071774  013701  100000                            MOV     @#100000,R1     ;READ FROM ADDRESS 100000
072000  020001                                    CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
072002  001401                                    BEQ     6$              ;BRANCH IF STORE WAS CORRECT
072004  104121                                    ERROR   121             ;INCORRECT STORE
072006                                  6$:       ;THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE
                                                ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
                                                ;THE  * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                                ;INTO THE FLIP/FLOPS ON SSRB.
                                                ;       D67.80    (117)
                                                ;       D67.00    (006)
                                                ;       D67.10    (251)
                                                ;   *   D10.30    (122)
                                                ;   *   D10.40    (157)
                                                ;       FET.01    (331)
072006  012737  072030  001110                    MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
072014  012737  010340  177776                    MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
072022  012700  052525                            MOV     #52525,R0       ;LOAD TEST DATA INTO R0
072026  005002                                    CLR     R2              ;MAKE REGISTER 2 ZERO
```

```
 9596  072030  010046                  16$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
 9597  072032  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 9598  072036  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9599  072040  006662  100000                  MTPI    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 100000
 9600  072044  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
 9601  072052  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 100000
 9602  072056  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
 9603  072060  001401                          BEQ     7$              ;BRANCH IF STORE WAS CORRECT
 9604  072062  104121                          ERROR   121             ;INCORRECT STORE
 9605  072064  012737  071516  001110  7$:     MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
 9606  072072  012737  025054  000250          MOV     #MMTRAP,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
 9607  072100  042737  000002  172516          BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
 9608  072106  000413                          BR      TST112          ;;BRANCH TO NEXT TEST
 9609
 9610
 9611  072110  013737  177572  001246  10$:     MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
 9612  072116  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
 9613  072124  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
 9614  072132  104117                          ERROR   117             ;TRIED TO LOAD A N.R. PAGE 4
 9615  072134  000002                          RTI                     ;RETURN TO TEST
 9616
 9617
 9618
 9619                                  ;;**********************************************************************
 9620                                  ;*TEST 112      MOVE FROM PREVIOUS (USER) I-SPACE
 9621                                  ;*
 9622                                  ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS USER THE
 9623                                  ;*      'USER SPACE (1) L'' FLIP-FLOP IS SET AND THE FETCH IS FROM
 9624                                  ;*      USER MODE.
 9625                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
 9626                                  ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
 9627                                  ;*      THEIR ADDRESSES).
 9628                                  ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
 9629                                  ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
 9630                                  ;*
 9631                                  ;;**********************************************************************
 9632  072136                          TST112:
 9633  072136  000004                          SCOPE
 9634  072140  012737  072530  001314          MOV     #TST113,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
 9635                                                                  ;TEST FOR ESCAPE ON PARITY ERRORS
 9636  072146  012700  077400          20$:    MOV     #77400,R0       ;MAKE PAGE 4 NON-RESIDENT IN ALL MODES
 9637                                                                  ;EXCEPT KERNEL I-SPACE & USER I-SPACE
 9638  072152  010037  172330                  MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
 9639  072156  010037  177630                  MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
 9640  072162  010037  172230                  MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
 9641  072166  010037  172210                  MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
 9642  072172  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
 9643  072200  012737  077406  177610          MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
 9644  072206  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
 9645  072214  012737  001000  177650          MOV     #1000,UIPAR4    ;MAP USER I PAGE 4 TO 16K
 9646  072222  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
 9647  072226  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 100000
 9648  072232  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9649  072236  012737  072502  000250          MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
 9650  072244  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
 9651  072250  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
```

```
 9652  072256  006506                  1$:     MFPI    USP                 ;PUT USER STACK POINTER ON KERNEL STACK
 9653  072260  022706  001100                  CMP     #KERSTK,KSP         ;WAS SOMETHING PUSHED ON STACK AT 1$
 9654  072264  001407                          BEQ     3$                  ;BRANCH IF NOTHING WAS PUSHED
 9655  072266  012601                          MOV     (KSP)+,R1           ;POP KERNEL STACK INTO R1
 9656  072270  012702  000600                  MOV     #USESTK,R2          ;EXPECTING 600 AS USP
 9657  072274  020201                          CMP     R2,R1               ;DID YOU GET THE RIGHT POINTER?
 9658  072276  001403                          BEQ     2$                  ;BRANCH IF YOU DID
 9659  072300  104114                          ERROR   114                 ;WRONG THING WAS PUSHED ON STACK
 9660  072302  000401                          BR      2$                  ;BRANCH TO NEXT TRY
 9661  072304  104115                  3$:     ERROR   115                 ;NOTHING WAS PUSHED ON THE STACK
 9662  072306                          2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE
 9663                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9664                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9665                                          ;INTO THE F/F'S ON SSRB.
 9666                                          ;   *  D12.00    (001)
 9667                                          ;   *  D12.10    (175)
 9668                                          ;   *  MFP.00    (066)
 9669                                          ;      MFP.10    (250)
 9670                                          ;      SVC.80    (222)
 9671                                          ;      SVC.90    (300)
 9672                                          ;      FET.00    (217)
 9673  072306  012737  072314  001110          MOV     #12$,$LPERR         ;SET LOOP ON ERROR POINTER TO 12$
 9674  072314  012737  030340  177776  12$:    MOV     #030340,PSW         ;MAKE PREVIOUS MODE USER
 9675  072322  012702  100000                  MOV     #100000,R2          ;PUT VIRTUAL ADDRESS IN R2
 9676  072326  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
 9677  072330  006512                          MFPI    (R2)                ;READ FROM PHYSICAL 100000
 9678  072332  012601                          MOV     (KSP)+,R1           ;POP KERNEL STACK INTO R1
 9679  072334  020001                          CMP     R0,R1               ;WAS DATA FETCHED SAME AS STORED
 9680  072336  001401                          BEQ     4$                  ;BRANCH IF CORRECT DATA WAS FETCHED
 9681  072340  104116                          ERROR   116                 ;WRONG DATA WAS FETCHED
 9682  072342                          4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
 9683                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9684                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9685                                          ;INTO THE F/F'S ON SSRB.
 9686                                          ;   *  D12.01    (002)
 9687                                          ;   *  D12.10    (175)
 9688                                          ;   *  MFP.00    (066)
 9689                                          ;      MFP.10    (250)
 9690                                          ;      SVC.80    (222)
 9691                                          ;      SVC.90    (300)
 9692                                          ;      FET.00    (217)
 9693  072342  012737  072350  001110          MOV     #14$,$LPERR         ;SET LOOP ON ERROR POINTER TO 14$
 9694  072350  012737  030340  177776  14$:    MOV     #030340,PSW         ;MAKE PREVIOUS MODE USER
 9695  072356  012702  100000                  MOV     #100000,R2          ;PUT VIRTUAL ADDRESS IN R2
 9696  072362  000240                          NOP                         ;THIS IS A SYNC POINT FOR SCOPING
 9697  072364  006522                          MFPI    (R2)+               ;READ FROM PHYSICAL 100000
 9698  072366  012601                          MOV     (KSP)+,R1           ;POP KERNEL STACK INTO R1
 9699  072370  020001                          CMP     R0,R1               ;WAS DATA FETCHED SAME AS STORED
 9700  072372  001401                          BEQ     5$                  ;BRANCH IF CORRECT DATA WAS FETCHED
 9701  072374  104116                          ERROR   116                 ;WRONG DATA WAS FETCHED
 9702  072376                          5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.  BELOW ARE THE
 9703                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9704                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9705                                          ;INTO THE F/F'S ON SSRB.
 9706                                          ;      D30.00    (003)
 9707                                          ;      D30.10    (221)
```

```
9708                                            ;     D10.20    (233)
9709                                            ;  *  D10.50    (311)
9710                                            ;  *  D10.60    (177)
9711                                            ;  *  MFP.00    (066)
9712                                            ;     MFP.10    (250)
9713                                            ;     SVC.80    (222)
9714                                            ;     SVC.90    (300)
9715                                            ;     FET.00    (217)
9716  072376  012737  072404  001110            MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
9717  072404  012737  030340  177776  15$:      MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
9718  072412  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
9719  072414  006537  100000                    MFPI    @#100000        ;READ FROM PHYSICAL 100000
9720  072420  012601                            MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
9721  072422  020001                            CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
9722  072424  001401                            BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
9723  072426  104116                            ERROR   116             ;WRONG DATA WAS FETCHED
9724  072430                            6$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.  BELOW ARE THE
9725                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9726                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
```

```
                                        ;INTO THE F/F'S ON SSRB.
                                        ;       D45.00    (004)
                                        ;   *   D10.30    (122)
                                        ;   *   D10.60    (177)
                                        ;   *   MFP.00    (066)
                                        ;       MFP.10    (250)
                                        ;       SVC.80    (222)
                                        ;       SVC.90    (300)
                                        ;       FET.00    (217)
072430  012737  072436  001110        MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
072436  012737  030340  177776  16$:   MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
072444  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
072450  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
072452  006542                        MFPI    -(R2)           ;READ FROM PHYSICAL 100000
072454  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
072456  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
072460  001401                        BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
072462  104116                        ERROR   116             ;WRONG DATA WAS FETCHED
072464  012737  025054  000250  7$:   MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
072472  012737  072146  001110        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
072500  000413                        BR      TST113          ;;BRANCH TO NEXT TEST

072502  013737  177572  001246  10$:   MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
072510  013737  177574  001250        MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
072516  013737  177576  001252        MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
072524  104117                        ERROR   117             ;TRIED TO READ NON-RESIDENT PAGE
072526  000002                        RTI                     ;RETURN TO TEST

                                    ;;**********************************************************************
                                    ;*TEST 113      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO SUPERVISOR MODE
                                    ;*
                                    ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
                                    ;*      'KERNEL SPACE (1) L'' FLIP-FLOP IS SET AND THE FETCH IS FROM
                                    ;*      KERNEL MODE.
                                    ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                    ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                    ;*      THEIR ADDRESSES).
                                    ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                    ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                    ;*
                                    ;;**********************************************************************
072530                              TST113:
072530  000004                              SCOPE
072532  012737  073174  001314              MOV     #TST114,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                    ;TEST FOR ESCAPE ON PARITY ERRORS
072540  012700  077406                      MOV     #77406,R0       ;MAKE ALL SUPER I-SPACE PAGES RESIDENT
                                                                    ;READ/WRITE, LENGTH 200 BLOCKS
072544  012702  000010                      MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
072550  012701  172200                      MOV     #SIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
072554  010021                  19$:        MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
072556  077202                              SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
072560  012737  072574  001106              MOV     #20$,$LPADR     ;SET LOOP POINTER TO 20$
072566  012737  072574  001110              MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
072574  012737  040340  177776  20$:        MOV     #040340,PSW     ;GO TO SUPERVISOR MODE FOR THIS TEST
072602  012700  077400                      MOV     #77400,R0       ;MAKE PAGE 4 NON-RESIDENT IN ALL MODES
```

```
9783
9784  072606  010037  172330                    MOV     R0,KDPDR4       ;EXCEPT SUPER I-SPACE & KERNEL I-SPACE
9785  072612  010037  172230                    MOV     R0,SDPDR4       ;KERNEL D-SPACE PAGE 4
9786  072616  010037  177630                    MOV     R0,UDPDR4       ;SUPERVISOR D-SPACE PAGE 4
9787  072622  010037  177610                    MOV     R0,UIPDR4       ;USER D-SPACE PAGE 4
9788  072626  012737  077406  172310            MOV     #77406,KIPDR4   ;USER I-SPACE PAGE 4
9789  072634  012737  077406  172210            MOV     #77406,SIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
9790  072642  012737  001000  172350            MOV     #1000,KIPAR4    ;SUPER I-SPACE PAGE 4 READ/WRITE
9791  072650  012737  001000  172250            MOV     #1000,SIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
9792  072656  012700  036514                    MOV     #36514,R0       ;MAP SUPER I PAGE 4 TO 16K
9793  072662  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO R0
9794  072666  012702  100000                    MOV     #100000,R2      ;LOAD DATA PATTERN INTO PHY 100000
9795  072672  012737  073146  000250            MOV     #10$,MMVEC      ;LOAD VIRTUAL ADDRESS INTO R2
9796  072700  105037  172210                    CLRB    SIPDR4          ;SET M.M. VECTOR TO 10$
9797  072704  012737  040340  177776            MOV     #040340,PSW     ;MAKE SUPER I-SPACE PAGE 4 NON-RESIDENT
9798  072712  000240                            NOP                     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
9799  072714  006506                    1$:     MFPI    KSP             ;THIS IS A SYNC POINT FOR SCOPING
9800  072716  022706  000700                    CMP     #SUPSTK,SSP     ;PUT KERNEL STACK POINTER ON SUPER STACK
9801  072722  001407                            BEQ     3$              ;WAS SOMETHING PUSHED ON STACK AT 1$
9802  072724  012601                            MOV     (SSP)+,R1       ;BRANCH IF NOTHING WAS PUSHED
9803  072726  012702  001100                    MOV     #KERSTK,R2      ;POP SUPERVISOR STACK INTO R1
9804  072732  020201                            CMP     R2,R1           ;EXPECTING 1100 AS KSP
9805  072734  001403                            BEQ     2$              ;DID YOU GET THE RIGHT POINTER?
9806  072736  104114                            ERROR   114             ;BRANCH IF YOU DID
9807  072740  000401                            BR      2$              ;WRONG THING WAS PUSHED ON STACK
9808  072742  104115                    3$:     ERROR   115             ;BRANCH TO NEXT TRY
9809  072744                            2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE    ;NOTHING PUSHED ON STACK
9810                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9811                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9812                                            ;INTO THE F/F'S ON SSRB.
9813                                            ;   *  D12.00    (001)
9814                                            ;   *  D12.10    (175)
9815                                            ;   *  MFP.00    (066)
9816                                            ;      MFP.10    (250)
9817                                            ;      SVC.80    (222)
9818                                            ;      SVC.90    (300)
9819                                            ;      FET.00    (217)
9820  072744  012737  072752  001110            MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
9821  072752  012737  040340  177776    12$:    MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
9822  072760  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
9823  072764  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
9824  072766  006512                            MFPI    (R2)            ;READ FROM PHYSICAL 100000
9825  072770  012601                            MOV     (SSP)+,R1       ;POP SUPER STACK INTO R1
9826  072772  020001                            CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
9827  072774  001401                            BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
9828  072776  104116                            ERROR   116             ;WRONG DATA WAS FETCHED
9829  073000                            4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
9830                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9831                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9832                                            ;INTO THE F/F'S ON SSRB.
9833                                            ;   *  D12.01    (002)
9834                                            ;   *  D12.10    (175)
9835                                            ;   *  MFP.00    (066)
9836                                            ;      MFP.10    (250)
9837                                            ;      SVC.80    (222)
9838                                            ;      SVC.90    (300)
```

```
 9839                                                ;     FET.00    (217)
 9840  073000  012737  073006  001110               MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
 9841  073006  012737  040340  177776  14$:         MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
 9842  073014  012702  100000                       MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9843  073020  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9844  073022  006522                               MFPI    (R2)+           ;READ FROM PHYSICAL 100000
 9845  073024  012601                               MOV     (SSP)+,R1       ;POP SUPER STACK INTO R1
 9846  073026  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9847  073030  001401                               BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9848  073032  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
 9849  073034                             5$:       ;THE FOLLOWING WILL TEST DSTM=3 MFPI.  BELOW ARE THE
 9850                                               ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9851                                               ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9852                                               ;INTO THE F/F'S ON SSRB.
 9853                                               ;     D30.00    (003)
 9854                                               ;     D30.10    (221)
 9855                                               ;     D10.20    (233)
 9856                                               ;   * D10.50    (311)
 9857                                               ;   * D10.60    (177)
 9858                                               ;   * MFP.00    (066)
 9859                                               ;     MFP.10    (250)
 9860                                               ;     SVC.80    (222)
 9861                                               ;     SVC.90    (300)
 9862                                               ;     FET.00    (217)
 9863  073034  012737  073042  001110               MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
 9864  073042  012737  040340  177776  15$:         MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
 9865  073050  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9866  073052  006537  100000                       MFPI    @#100000        ;READ FROM PHYSICAL 100000
 9867  073056  012601                               MOV     (SSP)+,R1       ;POP SUPERVISOR STACK INTO R1
 9868  073060  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9869  073062  001401                               BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9870  073064  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
 9871  073066                             6$:       ;THE FOLLOWING WILL TEST DSTM=4 MFPI.  BELOW ARE THE
 9872                                               ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
 9873                                               ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
 9874                                               ;INTO THE F/F'S ON SSRB.
 9875                                               ;     D45.00    (004)
 9876                                               ;   * D10.30    (122)
 9877                                               ;   * D10.60    (177)
 9878                                               ;   * MFP.00    (066)
 9879                                               ;     MFP.10    (250)
 9880                                               ;     SVC.80    (222)
 9881                                               ;     SVC.90    (300)
 9882                                               ;     FET.00    (217)
 9883  073066  012737  073074  001110               MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
 9884  073074  012737  040340  177776  16$:         MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
 9885  073102  012702  100002                       MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 9886  073106  000240                               NOP                     ;THIS IS A SYNC POINT FOR SCOPING
 9887  073110  006542                               MFPI    -(R2)           ;READ FROM PHYSICAL 100000
 9888  073112  012601                               MOV     (SSP)+,R1       ;POP SUPERVISOR STACK INTO R1
 9889  073114  020001                               CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 9890  073116  001401                               BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
 9891  073120  104116                               ERROR   116             ;WRONG DATA WAS FETCHED
 9892  073122  012737  025054  000250  7$:          MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
 9893  073130  012737  000340  177776               MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
 9894  073136  012737  072574  001110               MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
```

```
                                        BR      TST114          ;;BRANCH TO NEXT TEST

073146  013737  177572  001246  10$:    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
073154  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
073162  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
073170  104117                          ERROR   117             ;TRIED TO READ NON-RESIDENT PAGE
073172  000002                          RTI                     ;RETURN TO TEST

                                ;;**********************************************************************
                                ;*TEST 114      MFPD (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
                                ;*
                                ;*      THIS TEST CHECKS THAT ''SSRB IR15 L'' CAN INHIBIT THE ASSERTING
                                ;*      OF ''SSRB I SPACEB L'' SO THAT THE REFERENCE IS TO D-SPACE IF
                                ;*      THE INSTRUCTION IS MFPD (OR MTPD).  [THESE INSTRUCTIONS HAVE
                                ;*      BIT 15 SET IN THE I.R.]
                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
                                ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
                                ;*      THEIR ADDRESSES).
                                ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
                                ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
                                ;*
                                ;;**********************************************************************
073174                          TST114:
073174  000004                          SCOPE
073176  012737  073574  001314          MOV     #TST115,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
                                                                ;TEST FOR ESCAPE ON PARITY ERRORS
073204  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
                                                                ;READ/WRITE, LENGTH 200 BLOCKS
073210  012702  000010                  MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
073214  012701  172300                  MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
073220  010021                  19$:    MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
073222  077202                          SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
073224  012737  073240  001106          MOV     #20$,$LPADR     ;SET LOOP POINTER TO 20$
073232  012737  073240  001110          MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
073240  012700  077400          20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR D
                                                                ;AND KERNEL I NON-RESIDENT
073244  010037  172330                  MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
073250  010037  172210                  MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
073254  010037  177630                  MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
073260  010037  177610                  MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
073264  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
073272  012737  077406  172230          MOV     #77406,SDPDR4   ;SUPER D-SPACE PAGE 4 READ/WRITE
073300  012737  001000  172350          MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
073306  012737  001000  172270          MOV     #1000,SDPAR4    ;MAP SUPER D PAGE 4 TO 16K
073314  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
073320  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 100000
073324  012737  073546  000250          MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
073332  052737  000002  172516          BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
073340  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
073344                          2$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPD.  BELOW ARE THE
                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
                                        ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                        ;INTO THE F/F'S ON SSRB.
```

```
                                                ;  *  D12.00    (001)
                                                ;  *  D12.10    (175)
                                                ;  *  MFP.00    (066)
                                                ;     MFP.10    (250)
                                                ;     SVC.80    (222)
                                                ;     SVC.90    (300)
                                                ;     FET.00    (217)
073344  012737  073352  001110                  MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
073352  012737  010340  177776  12$:            MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
073360  012702  100000                          MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
073364  000240                                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
073366  106512                                  MFPD    (R2)            ;READ FROM PHYSICAL 100000
073370  012601                                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
073372  020001                                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
073374  001401                                  BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
073376  104116                                  ERROR   116             ;WRONG DATA WAS FETCHED
073400                                  4$:     ;THE FOLLOWING WILL TEST DSTM=2 MFPD.  BELOW ARE THE
                                                ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
                                                ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                                ;INTO THE F/F'S ON SSRB.
                                                ;  *  D12.01    (002)
                                                ;  *  D12.10    (175)
                                                ;  *  MFP.00    (066)
                                                ;     MFP.10    (250)
                                                ;     SVC.80    (222)
                                                ;     SVC.90    (300)
                                                ;     FET.00    (217)
073400  012737  073406  001110                  MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
073406  012737  010340  177776  14$:            MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
073414  012702  100000                          MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
073420  000240                                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
073422  106522                                  MFPD    (R2)+           ;READ FROM PHYSICAL 100000
073424  012601                                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
073426  020001                                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
073430  001401                                  BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
073432  104116                                  ERROR   116             ;WRONG DATA WAS FETCHED
073434                                  5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPD.  BELOW ARE THE
                                                ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
                                                ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
                                                ;INTO THE F/F'S ON SSRB.
                                                ;     D30.00    (003)
                                                ;     D30.10    (221)
                                                ;     D10.20    (233)
                                                ;  *  D10.50    (311)
                                                ;  *  D10.60    (177)
                                                ;  *  MFP.00    (066)
                                                ;     MFP.10    (250)
                                                ;     SVC.80    (222)
                                                ;     SVC.90    (300)
                                                ;     FET.00    (217)
073434  012737  073442  001110                  MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
073442  012737  010340  177776  15$:            MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
073450  000240                                  NOP                     ;THIS IS A SYNC POINT FOR SCOPING
073452  106537  100000                          MFPD    @#100000        ;READ FROM PHYSICAL 100000
073456  012601                                  MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
073460  020001                                  CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
```

```
10007  073462  001401                        BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
10008  073464  104116                        ERROR   116             ;WRONG DATA WAS FETCHED
10009  073466                          6$:   ;THE FOLLOWING WILL TEST DSTM=4 MFPD.  BELOW ARE THE
10010                                        ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10011                                        ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10012                                        ;INTO THE F/F'S ON SSRB.
10013                                        ;       D45.00    (004)
10014                                        ;   *   D10.30    (122)
10015                                        ;   *   D10.60    (177)
10016                                        ;   *   MFP.00    (066)
10017                                        ;       MFP.10    (250)
10018                                        ;       SVC.80    (222)
10019                                        ;       SVC.90    (300)
10020                                        ;       FET.00    (217)
10021  073466  012737  073474  001110        MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
10022  073474  012737  010340  177776  16$:  MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
10023  073502  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10024  073506  000240                        NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10025  073510  106542                        MFPD    -(R2)           ;READ FROM PHYSICAL 100000
10026  073512  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
10027  073514  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10028  073516  001401                        BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
10029  073520  104116                        ERROR   116             ;WRONG DATA WAS FETCHED
10030  073522  012737  025054  000250  7$:   MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
10031  073530  012737  073240  001110        MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
10032  073536  042737  000002  172516        BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
10033  073544  000413                        BR      TST115          ;;BRANCH TO NEXT TEST
10034
10035
10036  073546  013737  177572  001246  10$:  MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
10037  073554  013737  177574  001250        MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
10038  073562  013737  177576  001252        MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
10039  073570  104117                        ERROR   117             ;TRIED TO READ NON-RESIDENT PAGE
10040  073572  000002                        RTI                     ;RETURN TO TEST
10041
10042
10043                                        ;;*****************************************************************
10044                                        ;*TEST 115      MFPI (USER/PREV.USER) WITH USER D-SPACE ENABLED
10045                                        ;*
10046                                        ;*      THIS TEST CHECKS THAT, IF THE INSTRUCTION IS EITHER MFPI OR
10047                                        ;*      MTPI AND BOTH THE PRESENT AND PREVIOUS MODES ARE USER
10048                                        ;*      (PS=17XXXX), THEN ''SSRB I SPACEB L'' IS NOT ASSERTED AND
10049                                        ;*      D-SPACE IS USED IF IT WAS ENABLED.  [IN THIS WAY AN OPERATING
10050                                        ;*      SYSTEM CAN MAKE SOME PROPRIETARY CODE ''EXECUTE ONLY'' FOR
10051                                        ;*      THE USER.]
10052                                        ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
10053                                        ;*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
10054                                        ;*      THEIR ADDRESSES).
10055                                        ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
10056                                        ;*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
10057                                        ;*
10058                                        ;;*****************************************************************
10059  073574  000004                  TST115: SCOPE
10060  073576  012737  074210  001314        MOV     #TST116,NXTTST  ;SAVE STARTING ADDRESS OF NEXT TEST
10061  073604  012700  077406                MOV     #77406,R0       ;MAKE ALL USER I-SPACE PAGES RESIDENT
10062                                                                 ;READ/WRITE, LENGTH 200 BLOCKS
```

```
10063  073610  012702  000010                    MOV     #10,R2          ;SET COUNT TO LOAD 8 ADDRESSES
10064  073614  012701  177600                    MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
10065  073620  010021                  19$:      MOV     R0,(R1)+        ;LOAD R0 INTO PDR ADDRESSED BY R1
10066  073622  077202                            SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10067  073624  012737  073640  001106            MOV     #20$,$LPADR     ;SET LOOP POINTER TO 20$
10068  073632  012737  073640  001110            MOV     #20$,$LPERR     ;SET LOOP ON ERROR TO 20$
10069  073640  012737  077406  172310  20$:      MOV     #77406,KIPDR4   ;MAKE KERNEL I PAGE 4 R/W, 200 BLOCKS
10070  073646  012700  077400                    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT USER D
10071                                                                    ;AND USER I NON-RESIDENT
10072  073652  010037  172330                    MOV     R0,KDPDR4       ;KERNEL D-SPACE PAGE 4
10073  073656  010037  172210                    MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
10074  073662  010037  172230                    MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
10075  073666  012737  077406  177630            MOV     #77406,UDPDR4   ;USER D-SPACE PAGE 4 READ/WRITE
10076  073674  012737  001000  172350            MOV     #1000,KIPAR4    ;MAP KERNEL I PAGE 4 TO 16K
10077  073702  012737  001000  177650            MOV     #1000,UIPAR4    ;MAP USER I PAGE 4 TO 16K
10078  073710  012737  001000  177670            MOV     #1000,UDPAR4    ;MAP USER D PAGE 4 TO 16K
10079  073716  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
10080  073722  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 100000
10081  073726  012737  074162  000250            MOV     #10$,MMVEC      ;SET M.M. VECTOR TO 10$
10082  073734  052737  000001  172516            BIS     #BIT0,MMR3      ;ENABLE USER D-SPACE
10083  073742  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
10084  073746  105037  177610                    CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
10085  073752                           2$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE
10086                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10087                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10088                                            ;INTO THE F/F'S ON SSRB.
10089                                            ;   *  D12.00    (001)
10090                                            ;   *  D12.10    (175)
10091                                            ;   *  MFP.00    (066)
10092                                            ;      MFP.10    (250)
10093                                            ;      SVC.80    (222)
10094                                            ;      SVC.90    (300)
10095                                            ;      FET.00    (217)
10096  073752  012737  073760  001110                    MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
10097  073760  012737  170340  177776  12$:      MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10098  073766  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10099  073772  000240                            NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10100  073774  006512                            MFPI    (R2)            ;READ FROM PHYSICAL 100000
10101  073776  012601                            MOV     (USP)+,R1       ;POP USER STACK INTO R1
10102  074000  020001                            CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10103  074002  001401                            BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
10104  074004  104116                            ERROR   116             ;WRONG DATA WAS FETCHED
10105  074006                           4$:      ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
10106                                            ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10107                                            ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10108                                            ;INTO THE F/F'S ON SSRB.
10109                                            ;   *  D12.01    (002)
10110                                            ;   *  D12.10    (175)
10111                                            ;   *  MFP.00    (066)
10112                                            ;      MFP.10    (250)
10113                                            ;      SVC.80    (222)
10114                                            ;      SVC.90    (300)
10115                                            ;      FET.00    (217)
10116  074006  012737  074014  001110                    MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
10117  074014  012737  170340  177776  14$:      MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10118  074022  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
```

```
10119  074026  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10120  074030  006522                          MFPI    (R2)+           ;READ FROM PHYSICAL 100000
10121  074032  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
10122  074034  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10123  074036  001401                          BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
10124  074040  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
10125  074042                          5$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.  BELOW ARE THE
10126                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10127                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10128                                          ;INTO THE F/F'S ON SSRB.
10129                                          ;       D30.00    (003)
10130                                          ;       D30.10    (221)
10131                                          ;       D10.20    (233)
10132                                          ;   *   D10.50    (311)
10133                                          ;   *   D10.60    (177)
10134                                          ;   *   MFP.00    (066)
10135                                          ;       MFP.10    (250)
10136                                          ;       SVC.80    (222)
10137                                          ;       SVC.90    (300)
10138                                          ;       FET.00    (217)
10139  074042  012737  074050  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
10140  074050  012737  170340  177776  15$:    MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10141  074056  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10142  074060  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 100000
10143  074064  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
10144  074066  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10145  074070  001401                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS FETCHED
10146  074072  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
10147  074074                          6$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.  BELOW ARE THE
10148                                          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10149                                          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10150                                          ;INTO THE F/F'S ON SSRB.
10151                                          ;       D45.00    (004)
10152                                          ;   *   D10.30    (122)
10153                                          ;   *   D10.60    (177)
10154                                          ;   *   MFP.00    (066)
10155                                          ;       MFP.10    (250)
10156                                          ;       SVC.80    (222)
10157                                          ;       SVC.90    (300)
10158                                          ;       FET.00    (217)
10159  074074  012737  074102  001110          MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
10160  074102  012737  170340  177776  16$:    MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10161  074110  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10162  074114  000240                          NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10163  074116  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 100000
10164  074120  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
10165  074122  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10166  074124  001401                          BEQ     7$              ;BRANCH IF CORRECT DATA WAS FETCHED
10167  074126  104116                          ERROR   116             ;WRONG DATA WAS FETCHED
10168  074130  012737  025054  000250  7$:     MOV     #MMTRAP,MMVEC   ;SET M.M.VECTOR TO NORMAL ROUTINE
10169  074136  012737  073640  001110          MOV     #20$,$LPERR     ;SET LOOP POINTER TO START OF TEST
10170  074144  042737  000001  172516          BIC     #BIT0,MMR3      ;DISABLE USER D-SPACE
10171  074152  012737  000340  177776          MOV     #340,PSW        ;MAKE PRESENT MODE KERNEL
10172  074160  000413                          BR      TST116          ;;BRANCH TO NEXT TEST
10173
10174
```

```
10175  074162  013737  177572  001246  10$:    MOV     MMR0,PMMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
10176  074170  013737  177574  001250          MOV     MMR1,PMMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
10177  074176  013737  177576  001252          MOV     MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
10178  074204  104117                          ERROR   117             ;TRIED TO READ NON-RESIDENT PAGE
10179  074206  000002                          RTI                     ;RETURN TO TEST
10180
10181
10182
10183                                  ;;*****************************************************************
10184                                  ;*TEST 116      CHECK DPARS READ BACK CORRECTLY
10185                                  ;
10186                                  ;
10187                                  ;;*****************************************************************
10188  074210  000004                  TST116: SCOPE
10189  074212  013746  177776                  MOV     @#PS,-(SP)              ;CLEAR T BIT IF SET
10190  074216  042716  000020                  BIC     #20,(SP)                ;
10191  074222  012746  074230                  MOV     #18$,-(SP)              ;
10192  074226  000002                          RTI                             ;
10193  074230                          18$:                                    ;
10194  074230  012737  021022  001314          MOV     #$EOP,NXTTST            ;SET ESCAPE POINTER TO EOP ROUTINE
10195  074236  005037  177572                  CLR     MMR0                    ;TURN OFF MEM MGMT
10196  074242  005037  172516                  CLR     MMR3                    ;
10197  074246  005037  177776                  CLR     PS                      ;
10198  074252  012737  000014  177746          MOV     #14,@#CONTRL            ;TURN CACHE OFF
10199  074260  005037  172340                  CLR     KIPAR0                  ;
10200  074264  012737  177406  172300          MOV     #177406,KIPDR0          ;SET CONDITIONS FOR THIS TEST
10201  074272  012737  000200  172342          MOV     #200,KIPAR1             ;
10202  074300  012737  177406  172302          MOV     #177406,KIPDR1          ;
10203  074306  012737  000400  172344          MOV     #400,KIPAR2             ;
10204  074314  012737  177406  172304          MOV     #177406,KIPDR2          ;
10205  074322  012737  000600  172346          MOV     #600,KIPAR3             ;
10206  074330  012737  177406  172306          MOV     #177406,KIPDR3          ;
10207  074336  012737  001000  172350          MOV     #1000,KIPAR4            ;
10208  074344  012737  177406  172310          MOV     #177406,KIPDR4          ;
10209  074352  012737  001200  172352          MOV     #1200,KIPAR5            ;
10210  074360  012737  177406  172312          MOV     #177406,KIPDR5          ;
10211  074366  012737  001400  172354          MOV     #1400,KIPAR6            ;
10212  074374  012737  177406  172314          MOV     #177406,KIPDR6          ;
10213  074402  012737  007600  172356          MOV     #7600,KIPAR7            ;
10214  074410  012737  177406  172316          MOV     #177406,KIPDR7          ;
10215  074416  005037  172360                  CLR     KDPAR0                  ;
10216  074422  012737  177406  172320          MOV     #177406,KDPDR0          ;
10217  074430  012737  000200  172362          MOV     #200,KDPAR1             ;
10218  074436  012737  177406  172322          MOV     #177406,KDPDR1          ;
10219  074444  012737  000400  172364          MOV     #400,KDPAR2             ;
10220  074452  012737  177406  172324          MOV     #177406,KDPDR2          ;
10221  074460  012737  000600  172366          MOV     #600,KDPAR3             ;
10222  074466  012737  177406  172326          MOV     #177406,KDPDR3          ;
10223  074474  012737  001000  172370          MOV     #1000,KDPAR4            ;
10224  074502  012737  177406  172330          MOV     #177406,KDPDR4          ;
10225  074510  012737  001200  172372          MOV     #1200,KDPAR5            ;
10226  074516  012737  177406  172332          MOV     #177406,KDPDR5          ;
10227  074524  012737  001400  172374          MOV     #1400,KDPAR6            ;
10228  074532  012737  177406  172334          MOV     #177406,KDPDR6          ;
10229  074540  012737  007600  172376          MOV     #7600,KDPAR7            ;
10230  074546  012737  177406  172336          MOV     #177406,KDPDR7          ;
```

```
10231  074554  012737  000014  172516          MOV     #14,MMR3                ;SET FOR R/W
10232  074562  012737  074750  074764          MOV     #MAGIC,WORK             ;BEGIN WITH TEST DATA IN DPARS
10233  074570  022737  074764  074764  1$:     CMP     #WORK,WORK              ;TESTED ALL DPAR'S?
10234  074576  101461                          BLOS    7$                      ;BRANCH IF YES
10235  074600  012737  000001  177572          MOV     #1,MMR0                 ;TURN ON MEM MGMT
10236  074606  017703  000152                  MOV     @WORK,R3                ;GET ADDR OF FIRST DPAR
10237  074612  017700  000146                  MOV     @WORK,R0                ;
10238  074616  017701  000142                  MOV     @WORK,R1                ;
10239  074622  062701  000016                  ADD     #16,R1                  ;POINT TO LAST DPAR
10240  074626  062737  000002  074764          ADD     #2,WORK                 ;POINT TO TEST NUMBER
10241  074634  017702  000124                  MOV     @WORK,R2                ;GET TEST NUMBER
10242  074640  020103                  2$:     CMP     R1,R3                   ;ARE ALL DPAR'S LOADED?
10243  074642  101402                          BLOS    3$                      ;BRANCH IF YES
10244  074644  010223                          MOV     R2,(R3)+                ;LOAD DPAR USING KDPAR7
10245  074646  000774                          BR      2$                      ;DO DPAR0 THRU DPAR6
10246  074650  010003                  3$:     MOV     R0,R3                   ;POINT BACK TO DPAR0
10247  074652  020103                  4$:     CMP     R1,R3                   ;ARE ALL DPAR'S READ?
10248  074654  101421                          BLOS    5$                      ;
10249  074656  020213                          CMP     R2,(R3)                 ;READ TEST NUMBER BACK
10250  074660  001425                          BEQ     6$                      ;BRANCH IF COMPARE TRUE
10251  074662  011301                          MOV     (R3),R1                 ;GET THE RECEIVED DATA
10252  074664  005037  177572                  CLR     MMR0                    ;TURN OFF MEM MGMT
10253  074670  010137  001174                  MOV     R1,$TMP2                ;GET THE RECEIVED DATA
10254  074674  010337  001170                  MOV     R3,$TMP0                ;GET THE DPAR ADDRESS
10255  074700  010237  001172                  MOV     R2,$TMP1                ;GET THE TEST NUMBER
10256  074704  005037  172516                  CLR     MMR3                    ;DISABLE DPARS
10257  074710  104133                          ERROR   133                     ;DPAR DID NOT READ CORRECTLY
10258  074712  104400  012656                  TYPE,   ECO                     ;TYPE ECO MESSAGE
10259  074716  000423                          BR      DONE                    ;REPORT FIRST ERROR ONLY
10260  074720  005037  177572          5$:     CLR     MMR0                    ;TURN OFF MEM MGMT
10261  074724  062737  000002  074764          ADD     #2,WORK                 ;POINT TO NEXT PAR
10262  074732  000716                          BR      1$                      ;
10263  074734  062703  000002          6$:     ADD     #2,R3                   ;POINT TO NEXT DPAR
10264  074740  000744                          BR      4$                      ;
10265  074742  005037  172516          7$:     CLR     MMR3                    ;DISABLE DPARS
10266  074746  000407                          BR      DONE                    ;
10267
10268  074750  177660                  MAGIC:  .WORD   UDPAR0                  ;DPAR AND MAGIC NUMBER TABLE
10269  074752  007601                          .WORD   7601
10270  074754  172260                          .WORD   SDPAR0
10271  074756  007655                          .WORD   7655
10272  074760  172360                          .WORD   KDPAR0
10273  074762  007654                          .WORD   7654
10274  074764  000000                  WORK:   .WORD   0                       ;WORK LOCATION
10275
10276
10277  074766                          DONE:
10278  074766  000004                  END:    SCOPE                   ;LOOP BACK FOR LAST TEST
10279  074770  000240                          NOP                     ;THIS CAN BE ANYTHING YOU WANT
10280                                                                  ;(AS LONG AS IT IS ONLY ONE WORD)
10281  074772  000137  021022                  JMP     $EOP            ;JUMP TO END-OF-PASS ROUTINE
10282           000001                  .END
```

```
ABIT   = 000200  8096#  8126  8134
ADDROR   001274  1011#  2858  2862  2865  3882*  3915*  3941*  3971*
ADRAND   001276  1012#  2858  2862  2865  3886*  3917*  3943*  3973*
BADPC    001260  1005#  2837  2839  2842  2844   4018*  4065*  4115*  4159*  6319*  6469*
BIT0   = 000001   597#  6522  6991  7043  7591   7939   8093   9080  10082  10170
BIT00  = 000001   587#   597
BIT01  = 000002   586#   596
BIT02  = 000004   585#   595
BIT03  = 000010   584#   594
BIT04  = 000020   583#   593
BIT05  = 000040   582#   592
BIT06  = 000100   581#   591
BIT07  = 000200   580#   590
BIT08  = 000400   579#   589  3152
BIT09  = 001000   578#   588  3162  3235
BIT1   = 000002   596#  4074*  7836  9389  9476  9518  9607  9945  10032
BIT10  = 002000   577#  3218
BIT11  = 004000   576#  3169
BIT12  = 010000   575#  3106  6776  7075
BIT13  = 020000   574#  3225  4642
BIT14  = 040000   573#  4661  7332  7334  7376  7378
BIT15  = 100000   572#  4604  4645  7068  7116  7125
BIT2   = 000004   595#  7605  7743  8024  8045
BIT3   = 000010   594#  4815
BIT4   = 000020   593#  3821  6351  6393  6523  7311  7592  8094  9081
BIT5   = 000040   592#
BIT6   = 000100   591#  8095
BIT7   = 000200   590#  8096
BIT8   = 000400   589#  6013  6027  6041  6055  6069  6083  6097  6111  6125  6139  6153  6167
BIT9   = 001000   588#  1051  3287  6982  8122  8130
BPTVEC= 000014   604#
CACHVE= 000114   611#  4055*  4068*  4077*  4281*  4282*
CLEANU   024174  3880#  4285  4874  4906  4938  4970  5002  5034  5077  5139  5201  5263  5325
                 5387  5459  5502  5545  5587  5630  5673
CLRREG   024156  3863#  5080  5097  5142  5159  5204  5221  5266  5283  5328  5345  5390  5407
                 5987  5989  5991  5993  5995  5997
CONTRL= 177746   621#  4062  10198*
CPFLAG   024520  3248*  4008#  4027*  4271*
CPUER    024516  4007#  4279  4341  4884  4916  4948  4980  5012  5044  6245  6394
CPUERR= 177766   634#  3244*  3909  3910*  4017  4026*  4276*  6298  6323*  6448  6473*
CPUEXP   001222  988#  2836  3911  4019  4021  4277*  6249*  6270*  6281*  6290*  6395*  6419*  6430*
                 6439*
CR     = 000015  509#  3530  3540
CRLF   = 000200  510#  1680  1694  2059  2104  2439  2454  2466  2521  2576  2607  2674  2680
                 2697  2818  3504  3540  4294
DATAND   001266  1008#  2862  2865  3885*  3947*  3981*
DATAOR   001264  1007#  2862  2865  3881*  3945*  3979*
DF1      020466  1075  2951#
DF100    020733  1467  3019#
DF101    020737  1473  3021#
DF102    020743  1479  1485  1491  1497  1503  1509  1515  1521  3023#
DF11     020516  1127  2960#
DF112    020746  1527  3024#
DF113    020752  1533  3026#
DF114    020756  1539  3028#
DF116    020762  1551  1569  3030#
```

```
DF117   020766          1557    3032#
DF12    020522          1133    1151    2962#
DF120   020773          1563    3034#
DF122   020776          1575    1581    1587    1611    3035#
DF127   021001          1605    3036#
DF13    020525          1139    1145    2963#
DF131   021003          1617    3037#
DF15    020534          2966#
DF16    020531          1157    2965#
DF17    020540          1163    2968#
DF2     020467          1081    2952#
DF20    020544          1169    2970#
DF201   021010          1640    3042#
DF202   021012          1649    3043#
DF203   021015          1657    3044#
DF21    020550          1176    2972#
DF22    020556          1182    2974#
DF23    020566          1188    2977#
DF24    020573          1194    2979#
DF25    020576          1200    1206    1212    1218    2980#
DF3     020472          1088    1095    2953#
DF31    020602          1224    1230    1236    1242    1630    2982#
DF35    020607          1248    2984#
DF36    020612          1255    2985#
DF37    020617          1261    2987#
DF40    020624          1267    1593    2989#
DF41    020630          1273    1599    2991#
DF42    020633          1280    1305    1312    2992#
DF43    020640          1286    2994#
DF44    020642          1292    2995#
DF45    020645          1298    2996#
DF5     020500          1102    1624    2955#
DF50    020650          1318    2997#
DF51    020654          1324    1330    2999#
DF53    020660          1336    1342    1366    1384    1409    1424    1443    1455    1545    3001#
DF55    020662          1348    1360    1378    3002#
DF56    020666          1354    1372    3004#
DF6     020505          1109    2957#
DF64    020672          1391    3006#
DF65    020675          1397    3007#
DF66    020701          1403    3009#
DF7     020513          1115    1121    2959#
DF70    020704          1418    3010#
DF72    020712          1430    3012#
DF73    020716          1437    3014#
DF75    020726          1449    1461    3017#
DH1     013214          1073    2432#
DH100   016260          1465    2720#
DH101   016320          1471    2726#
DH102   016360          1477    1483    1489    1495    1501    1507    2732#
DH11    013670          1125    2485#
DH110   016410          1513    1519    2736#
DH112   016440          1525    2740#
DH113   016500          1531    2746#
DH114   016540          1537    2752#
DH116   016600          1549    2758#
```

```
DH117   016640          1555    2764#
DH12    013740          1131    2493#
DH120   016710          1561    2771#
DH121   016740          1567    2775#
DH122   017000          1573    1609    2781#
DH123   017051          1579    1585    2788#
DH127   017101          1603    2793#
DH13    013730          1137    2491#
DH131   017120          1615    2796#
DH133   017211          1628    2806#
DH14    013770          1143    2497#
DH15    014040          1149    2505#
DH16    014030          1155    2503#
DH17    014070          1161    2509#
DH2     013224          1079    2434#
DH20    014130          1167    2515#
DH201   017260          1636    2815#
DH202   017277          1644    2818#
DH203   017347          1653    2826#
DH21    014170          1173    2521#
DH22    014310          1180    2535#
DH23    014407          1186    2546#
DH24    014457          1192    2553#
DH25    014507          1198    1204    1210    1216    2558#
DH3     013260          1085    1092    2439#
DH31    014547          1222    1228    1234    1240    2564#
DH35    014617          1246    2571#
DH36    014647          1252    2576#
DH37    014747          1259    2588#
DH40    015017          1265    1591    2595#
DH41    015063          1271    1597    2602#
DH42    015115          1277    1302    1309    2607#
DH43    015217          1284    2620#
DH44    015237          1290    2623#
DH45    015267          1296    2628#
DH5     013410          1099    1621    2454#
DH50    015322          1316    2633#
DH51    015366          1322    1328    2639#
DH53    015426          1334    1340    1364    1382    1407    1422    1441    1453    1543    2645#
DH55    015446          1346    1358    1376    2648#
DH56    015506          1352    1370    2654#
DH6     013514          1106    2466#
DH64    015546          1389    2660#
DH65    015576          1395    2664#
DH66    015636          1401    2670#
DH7     013640          1113    1119    2481#
DH70    015666          1414    2674#
DH72    016017          1428    2691#
DH73    016057          1434    2697#
DH75    016210          1447    1459    2713#
DISPLA= 177570           504#   3183*   3214*   4329*   4582*   4873*   5985*   6511*   7580*   8082*   9069*
DONE    074766         10259   10266   10277#
DT1     017416          1074    2836#
DT100   020236          1466    2918#
DT101   020250          1472    2920#
DT102   020262          1478    1484    1490    1496    1502    1508    1514    1520    2922#
```

```
DT11    017510          1126    2849#
DT112   020272          1526    2924#
DT113   020304          1532    2926#
DT114   020316          1538    2928#
DT116   020330          1550    1568    2930#
DT117   020342          1556    2932#
DT12    017522          1132    1150    2851#
DT120   020356          1562    2934#
DT122   020366          1574    1580    1586    1610    2936#
DT127   020376          1604    2938#
DT13    017532          1138    1144    2853#
DT131   020404          1616    2939#
DT133   020420          1629    2941#
DT15    017546          2856#
DT16    017544          1156    2855#
DT17    017556          1162    2858#
DT2     017420          1080    2837#
DT20    017570          1168    2860#
DT201   020434          1639    2944#
DT202   020442          1648    2945#
DT203   020452          1656    2947#
DT21    017602          1175    2862#
DT22    017620          1181    2865#
DT23    017642          1187    2868#
DT24    017656          1193    2870#
DT25    017666          1199    1205    1211    1217    2872#
DT3     017430          1087    1094    2839#
DT31    017700          1223    1229    1235    1241    2874#
DT35    017714          1247    2876#
DT36    017724          1254    2878#
DT37    017740          1260    2880#
DT40    017754          1266    1592    2882#
DT41    017766          1272    1598    2884#
DT42    017776          1279    1304    1311    2886#
DT43    020012          1285    2888#
DT44    020020          1291    2889#
DT45    020030          1297    2891#
DT5     017446          1101    1623    2842#
DT50    020040          1317    2893#
DT51    020052          1323    1329    2895#
DT53    020064          1335    1341    1365    1383    1408    1423    1442    1454    1544    2897#
DT55    020072          1347    1359    1377    2898#
DT56    020104          1353    1371    2900#
DT6     017462          1108    2844#
DT64    020116          1390    2902#
DT65    020126          1396    2904#
DT66    020140          1402    2906#
DT7     017500          1114    1120    2847#
DT70    020150          1417    2908#
DT72    020166          1429    2911#
DT73    020200          1436    2913#
DT75    020222          1448    1460    2916#
DUALAD  024352          3939#   5107    5169    5231    5293    5355    5417
ECO     012656          2390#   10258
EMTVEC= 000030           607#   4251*   4252*
EM1     002736          1072    1661#
```

EM10	003567	1118	1736#
EM100	010374	1464	2172#
EM101	010445	1470	2179#
EM102	010512	1476	2186#
EM103	010553	1482	2192#
EM104	010634	1488	2201#
EM105	010703	1494	2208#
EM106	010760	1500	2216#
EM107	011041	1506	2225#
EM11	003620	1124	1741#
EM110	011110	1512	2232#
EM111	011157	1518	2239#
EM112	011246	1524	2249#
EM113	011301	1530	2254#
EM114	011345	1536	2261#
EM115	011413	1542	2268#
EM116	011474	1548	2277#
EM117	011542	1554	2284#
EM12	003662	1130	1747#
EM120	011607	1560	2291#
EM121	011664	1566	2299#
EM122	011727	1572	2305#
EM123	012011	1578	2314#
EM124	012101	1584	2324#
EM125	012155	1590	2332#
EM126	012236	1596	2341#
EM127	012320	1602	2350#
EM13	003736	1136	1755#
EM130	012410	1608	2360#
EM131	012476	1614	2369#
EM132	012557	1620	2378#
EM133	012623	1627	2385#
EM14	003772	1142	1760#
EM15	004026	1148	1765#
EM16	004102	1154	1773#
EM17	004135	1160	1778#
EM2	003026	1078	1671#
EM20	004203	1166	1785#
EM201	012757	1635	2402#
EM202	013034	1643	2410#
EM203	013125	1652	2420#
EM21	004244	1172	1791#
EM22	004306	1179	1797#
EM23	004350	1185	1803#
EM24	004414	1191	1809#
EM25	004453	1197	1815#
EM26	004531	1203	1823#
EM27	004577	1209	1830#
EM3	003107	1084	1680#
EM30	004636	1215	1836#
EM31	004725	1221	1846#
EM32	004766	1227	1852#
EM33	005052	1233	1861#
EM34	005131	1239	1869#
EM35	005215	1245	1878#
EM36	005303	1251	1888#

```
EM37    005352          1258    1895#
EM4     003225          1091    1694#
EM40    005431          1264    1903#
EM41    005512          1270    1912#
EM42    005574          1276    1921#
EM43    005645          1283    1928#
EM44    005723          1289    1936#
EM45    005772          1295    1943#
EM46    006047          1301    1951#
EM47    006130          1308    1960#
EM5     003350          1098    1710#
EM50    006163          1315    1965#
EM51    006231          1321    1972#
EM52    006304          1327    1980#
EM53    006361          1333    1988#
EM54    006433          1339    1996#
EM55    006502          1345    2003#
EM56    006565          1351    2012#
EM57    006644          1357    2020#
EM6     003423          1105    1718#
EM60    006726          1363    2029#
EM61    007007          1369    2038#
EM62    007066          1375    2046#
EM63    007150          1381    2055#
EM64    007200          1387    2059#
EM65    007343          1394    2077#
EM66    007413          1400    2084#
EM67    007474          1406    2093#
EM7     003513          1112    1728#
EM70    007575          1412    2104#
EM71    007660          1421    2113#
EM72    007745          1427    2122#
EM73    010026          1433    2131#
EM74    010130          1440    2142#
EM75    010202          1446    2149#
EM76    010255          1452    2157#
EM77    010321          1458    2164#
END     074766         10278#
ENMMTR  001354          1051#   7066
ENTPT2  041436          1044    4578#
ENTPT3  042614          1045    4869#
ENTPT4  047236          1046    5981#
ENTPT5  052274          1047    6507#
ENTPT6  057446          1048    7576#
ENTPT7  062136          1049    8078#
ENTPT8  067636          1050    9065#
EREXIT  022066          3243    3251#
ERRCNT  001300          1013#   2858    2862    2865    3137*   3205*   4885    4887    4917    4919    4949    4951    4981
                        4983    5013    5015    5045    5047    5116    5118    5178    5180    5240    5242    5302    5304
                        5364    5366    5426    5428    5478    5480    5521    5523    5564    5566    5607    5609    5650
                        5652    5693    5695
ERRVEC= 000004           600#   3143    3144*   3146*   3149*   4279*   4280*   4333*   4341*   4875*   4884*   4907*   4916*
                        4939*   4948*   4971*   4980*   5003*   5012*   5035*   5044*   6194*   6245*   6352*   6394*   7312*
                        7379*
ERTYPE  022070          3227    3277#
ER200   002706          1632#   3297    3300
```

```
FSTTST  001220          986#  4225*  4227*  4229*  4231*  4233*  4235*  4237*  4239*  4295
GNS   = ****** U        887   3075   3082   3711   3712   3713   3714   3715   3716   3717   3718   3719   4293
HIADRS= 177742          619#  4060
HITMIS= 177752          623#  4064
HOLFLG  001302         1014#  6193*  6210   6215*  6228   6314   6317*  6322*  6353*  6364   6369*  6382   6464
                       6467*  6472*
HT    = 000011          507#  3502   3540
IOTVEC= 000020          605#  4249*  4250*
KDPAR0= 172360          777#  7599*  7738*  8025*  8031*  8208*  10215* 10272
KDPAR1= 172362          778#  7600*  8209*  10217*
KDPAR2= 172364          779#  8210*  10219*
KDPAR3= 172366          780#  8211*  10221*
KDPAR4= 172370          781#  7601*  7739*  8023*  8212*  10223*
KDPAR5= 172372          782#  8213*  10225*
KDPAR6= 172374          783#  8214*  10227*
KDPAR7= 172376          784#  4881   5098   5113   5471   7603*  7741*  8215*  10229*
KDPDR0= 172320          755#  7594*  7733*  8277   8394   8535   8654   8681   8685*  8688*  8701*  8704*  8721*
                       8724*  8741*  8744*  8800   8820*  8836*  8856*  8876*  8943   8955*  8971*  8991*  9011*
                      10216*
KDPDR1= 172322          756#  7595*  7734*  8686*  8702*  8722*  8742*  8821*  8837*  8857*  8877*  8956*  8972*
                       8992*  9012*  10218*
KDPDR2= 172324          757#  7597*  7736*  10220*
KDPDR3= 172326          758#  7598*  7737*  10222*
KDPDR4= 172330          759#  7596*  7735*  8022*  9093*  9232*  9378*  9509*  9638*  9784*  9934*  10072* 10224*
KDPDR5= 172332          760#  10226*
KDPDR6= 172334          761#  10228*
KDPDR7= 172336          762#  4977   5284   5299   5600   7602*  7740*  8687*  8689*  8703*  8705*  8723*  8725*
                       8743*  8745*  8822*  8838*  8858*  8878*  8957*  8973*  8993*  9013*  10230*
KERSTK= 001100          494#  4248   4286   4539   4546   7020   8032   9110   9653   9803
KERVEC  025304         4192#  7275   7465
KIPAR0= 172340          766#  3402   4877   5079   5081   5095   5096   5110   5462   5722   5723*  5726   5731
                       5735   5988   6003*  6517*  7586*  8088*  8200*  9075*  10199*
KIPAR1= 172342          767#  6004*  6518*  7587*  8089*  8201*  9076*  10201*
KIPAR2= 172344          768#  6005*  6519*  7588*  8090*  8202*  9077*  10203*
KIPAR3= 172346          769#  6006*  6520*  7589*  8091*  8203*  9078*  10205*
KIPAR4= 172350          770#  2893   6010*  6024*  6038*  6052*  6066*  6080*  6094*  6108*  6122*  6136*  6150*
                       6164*  6195*  6223*  6247*  6271*  6283*  6301   6356*  6378*  6396*  6420*  6432*  6451
                       6525*  6578*  6632*  6685   6708*  6753*  6816*  6871*  6920*  7016*  7063*  7732*  8098*
                       8139*  8204*  9099*  9238*  9384*  9515*  9644*  9790*  9940*  10076* 10207*
KIPAR5= 172352          771#  6196*  6212   6224*  6226   6316   6357*  6366   6379*  6380   6397*  6466   6633*
                       6675*  6707*  6752*  6815*  6870*  6919*  7017*  8205*  10209*
KIPAR6= 172354          772#  8206*  10211*
KIPAR7= 172356          773#  6007*  6521*  6961   7590*  8092*  8207*  9079*  10213*
KIPDR0= 172300          744#  4973   5265   5267   5281   5282   5296   5590   5827   5828*  5831   5836   5840
                       5986   6000   6512*  7581*  8083*  8254   8281   8285*  8288*  8293   8301*  8304*  8321*
                       8324*  8341*  8344*  8398   8418*  8426   8434*  8454*  8474*  8539   8551*  8559   8567*
                       8587*  8607*  8677   8693   8796   8828   8939   8963   9070*  9086   9371   9927   10200*
KIPDR1= 172302          745#  6513*  7582*  8084*  8286*  8302*  8322*  8342*  8419*  8435*  8455*  8475*  8552*
                       8568*  8588*  8608*  9071*  10202*
KIPDR2= 172304          746#  6514*  7583*  8085*  9072*  10204*
KIPDR3= 172306          747#  6515*  7584*  8086*  9073*  10206*
KIPDR4= 172310          748#  2900   6524*  6526   6577*  6579   6629*  6712*  6757*  6820*  6875*  6924*  7019*
                       7064*  7154*  7593*  7731*  7747*  7782*  7784*  8097*  8103   8110   8116*  8117   8125
                       8133   8140   8153*  9097*  9104*  9236*  9264*  9267*  9286*  9289*  9307*  9310*  9330*
                       9333*  9382*  9390*  9513*  9531*  9534*  9553*  9556*  9574*  9577*  9597*  9600*  9642*
                       9650*  9788*  9938*  9946*  10069* 10083* 10208*
```

```
KIPDR5= 172312            749#   6630*  6662   6664   6666*  6668*  6709*  6754*  6817*  6872*  6921*  7018*  10210*
KIPDR6= 172314            750#  10212*
KIPDR7= 172316            751#   6516*  6959*  6964   6992*  7585*  8087*  8147   8155   8287*  8289*  8303*  8305*
                         8323*   8325*  8343*  8345*  8420*  8436*  8456*  8476*  8553*  8569*  8589*  8609*  9074*
                        10214*
LF    = 000012            508#   3534   3540
LOADRS= 177740            618#   4059
LOOP    040264           3116    4268#
MAGIC   074750          10232    10268#
MAINT = 177750            622#   4063
MAPH0 = 170202            861#
MAPH00= 170202            797#    861
MAPH01= 170206            799#    863
MAPH02= 170212            801#    865
MAPH03= 170216            803#    867
MAPH04= 170222            805#    869
MAPH05= 170226            807#    871
MAPH06= 170232            809#    873
MAPH07= 170236            811#    875
MAPH1 = 170206            863#
MAPH10= 170242            813#
MAPH11= 170246            815#
MAPH12= 170252            817#
MAPH13= 170256            819#
MAPH14= 170262            821#
MAPH15= 170266            823#
MAPH16= 170272            825#
MAPH17= 170276            827#
MAPH2 = 170212            865#
MAPH20= 170302            829#
MAPH21= 170306            831#
MAPH22= 170312            833#
MAPH23= 170316            835#
MAPH24= 170320            837#
MAPH25= 170326            839#
MAPH26= 170332            841#
MAPH27= 170336            843#
MAPH3 = 170216            867#
MAPH30= 170342            845#
MAPH31= 170346            847#
MAPH32= 170352            849#
MAPH33= 170356            851#
MAPH34= 170362            853#
MAPH35= 170366            855#
MAPH36= 170372            857#
MAPH37= 170376            859#
MAPH4 = 170222            869#
MAPH5 = 170226            871#
MAPH6 = 170232            873#
MAPH7 = 170236            875#
MAPL0 = 170200            860#   6957*  6974   6985   8146*
MAPL00= 170200            796#    860
MAPL01= 170204            798#    862
MAPL02= 170210            800#    864
MAPL03= 170214            802#    866
MAPL04= 170220            804#    868
```

```
MAPL05= 170224            806#    870
MAPL06= 170230            808#    872
MAPL07= 170234            810#    874
MAPL1 = 170204            862#
MAPL10= 170240            812#
MAPL11= 170244            814#
MAPL12= 170250            816#
MAPL13= 170254            818#
MAPL14= 170260            820#
MAPL15= 170264            822#
MAPL16= 170270            824#
MAPL17= 170274            826#
MAPL2 = 170210            864#
MAPL20= 170300            828#
MAPL21= 170304            830#
MAPL22= 170310            832#
MAPL23= 170314            834#
MAPL24= 170320            836#
MAPL25= 170324            838#
MAPL26= 170330            840#
MAPL27= 170334            842#
MAPL3 = 170214            866#
MAPL30= 170340            844#
MAPL31= 170344            846#
MAPL32= 170350            848#
MAPL33= 170354            850#
MAPL34= 170360            852#
MAPL35= 170364            854#
MAPL36= 170370            856#
MAPL37= 170374            858#
MAPL4 = 170220            868#
MAPL5 = 170224            870#
MAPL6 = 170230            872#
MAPL7 = 170234            874#
MEMER   024622           4047#   4068    4077    4281
MEMERR= 177744            620#   3245*   4061    4086*   4275*
MMEXP   001224            989#   2844    2904    4121    4125    4278*   6535*   6539*   6588*   6592*   6639*   6644*   6652*
                         6657*   6678*   6689*   6713*   6718*   6721*   6760*   6763*   6784*   6788*   6826*   6831*   6841*
                         6845*   6878*   6890*   6894*   6925*   6958*   6973*   6975*   7035*   7037    7044*   7069*   7079*
                         7091*   7218*   7408*
MMFLAG  025056           3250*   4108#   4129*   4273*
MMR0  = 177572            645#    649    2900    3246*   4118    4128*   4162    4170*   4212    4268*   4583    4645*   4659
                         4668*   4670    4681    4693    4709    4723    4734*   4754    4757*   6013*   6027*   6041*   6055*
                         6069*   6083*   6097*   6111*   6125*   6139*   6153*   6167*   6201*   6522*   6775    6783*   6828*
                         6840*   6881*   6889*   6928*   6931*   6960*   6967    6971*   6982*   6991*   7022*   7034    7043*
                         7071*   7074    7082*   7085*   7123    7134*   7170    7180    7195*   7283*   7358    7381*   7473*
                         7520    7532*   7591*   8019*   8034    8044*   8093*   8122*   8130*   8163*   9080*   9207    9343
                         9480    9611    9750    9898   10036   10175   10195*  10235*  10252*  10260*
MMR1  = 177574            646#    650    4119    4163    4213    4641    6968    7171    7181    7359    8035    9208    9344
                         9481    9612    9751    9899   10037   10176
MMR2  = 177576            647#    651    4120    4164    4214    4755    4766    6969    7172    7182    7360    7519    8036
                         9209    9345    9482    9613    9752    9900   10038   10177
MMR3  = 172516            648#    652    4269*   4802    6351*   6393*   6523*   6970    7592*   7604    7742    7835    7938
                         8024*   8045*   8094*   8653*   8762*   8788*   8897*   8923*   9032*   9081*   9082*   9389*   9476*
                         9518*   9607*   9945*  10032*  10082*  10170*  10196*  10231*  10256*  10265*
MMTRAP  025054           4107#   4283    7042    7196    7267    7382    7457    7535    8046    8160    9202    9339    9474
```

```
                    9606    9745    9892   10030   10168
MMVEC = 000250       613#   4283*   4284*   7015*   7042*   7113*   7156*   7157*   7162*   7196*   7267*   7275*   7276*
                    7324*   7382*   7457*   7465*   7466*   7506*   7507*   7535*   7607*   8046*   8047*   8099*   8160*
                    9103*   9202*   9240*   9339*   9388*   9474*   9517*   9606*   9649*   9745*   9795*   9892*   9944*
                   10030*  10081*  10168*
NDFLAG 025176       4152#   4171*   4274*
NODSPA 025174       4150#   7607
NXTTST 001314       1024#   4084    4330*   4361*   4403*   4427*   4454*   4482*   4508*   4536*   4576*   4639*   4752*
                    4800*   4867*   4904*   4936*   4968*   5000*   5032*   5075*   5137*   5199*   5261*   5323*   5385*
                    5456*   5499*   5542*   5584*   5627*   5670*   5718*   5753*   5788*   5823*   5858*   5893*   5933*
                    5974*   6191*   6343*   6504*   6574*   6627*   6704*   6749*   6812*   6867*   6916*   6953*   7012*
                    7061*   7111*   7152*   7214*   7309*   7406*   7491*   7572*   7729*   7810*   7913*   8010*   8074*
                    8194*   8383*   8516*   8649*   8784*   8919*   9061*   9228*   9366*   9505*   9634*   9771*   9922*
                   10060*  10194*
OLDPC  001304       1016#   3907*   3926    3940*   3954    3970*   3988    4015*   4018    4029    4057*   4065    4082*
                    4084*   4089    4116*   4131    4160*   4173    4193*   4200*   4206*   4216    6296*   6304*   6319
                    6321*   6325    6446*   6454*   6469    6471*   6475
OLDPS  001306       1018#   3908*   3925    4016*   4028    4058*   4088    4117*   4130    4161*   4172    4194*   4201*
                    4207*   4215    6297*   6324    6447*   6474
OLDPSW 001310       1020#   3825*   3840    3841*
PADRSH 001230        992#   3361*   3408*
PADRSL 001226        990#   3362*   3363    3409*   3410
PAFLAG 024624       3249*   4048#   4087*   4272*
PARCOU 024426       3969#   5469    5512    5555    5598    5641    5684
PARTAB 001320       1033#   5939*   5945    5948
PATAND 001272       1010#   2865    3887*   3977*
PATTOR 001270       1009#   2865    3883*   3975*
PCONTR 001240        997#   2839    4062*
PCPUER 001256       1004#   2837    3909*   3910    3911    4017*   4021    6206*   6208    6250*   6255    6282*   6286
                    6298*   6299    6312    6360*   6362    6431*   6435    6448*   6449    6462
PHIADR 001234        995#   4060*
PHITMI 001244        999#   4064*
PIRQ   = 177772      502#
PIRQVE= 000240       612#
PLOADR 001232        994#   2839    4059*   4072
PMAINT 001242        998#   2839    4063*
PMMR0  001246       1000#   2842    2844    2904    2906    2908    2911    2913    2916    2920    2932    2939    4118*
                    4125    4162*   4165    4212*   6528*   6540    6581*   6593    6759*   6764    6775*   6776    6782*
                    6789    6825*   6832    6839*   6846    6888*   6895    6972*   6976    6983*   6988    7034*   7037
                    7074*   7075    7083*   7088    7123*   7130    7170*   7189    7279    7358*   7361    7469    7520*
                    7529    8034*   8040    9207*   9343*   9480*   9611*   9750*   9898*  10036*  10175*
PMMR1  001250       1001#   2842    2844    2908    2913    2916    2932    2939    4119*   4163*   4213*   7171*   7186
                    7359*   7365    8035*   9208*   9344*   9481*   9612*   9751*   9899*  10037*  10176*
PMMR2  001252       1002#   2842    2844    2908    2913    2916    2918    2932    2939    4120*   4164*   4214*   7172*
                    7183    7360*   7369    7519*   7521    8036*   9209*   9345*   9482*   9613*   9752*   9900*  10038*
                   10177*
PMMR3  001254       1003#
PPARER 001236        996#   2839    4061*   4066    4086
PR0    = 000000      534#
PR1    = 000040      535#
PR2    = 000100      536#
PR3    = 000140      537#
PR4    = 000200      538#
PR5    = 000240      539#
PR6    = 000300      540#
PR7    = 000340      541#
```

```
PS     = 177776          499#   500    3104   10189  10197*
PSW    = 177776          500#  4192    4202    4208   4297*  4299*  4301*  4538*  4540*  4542*  4544*  4707*  4721*
                        4733*  7116*   7245*   7250*  7284*  7332*  7334*  7376*  7378*  7435*  7440*  7474*  7837*
                        7895*  7940*   7995*   8028   8030   8253*  8362*  8386*  8495*  8519*  8643*  8652*  8763*
                        8787*  8898*   8922*   9033*  9106*  9131*  9151*  9174*  9194*  9241*  9249*  9283*  9303*
                        9326*  9403*   9423*   9446*  9466*  9550*  9570*  9593*  9651*  9674*  9694*  9717*  9737*
                        9781*  9797*   9821*   9841*  9864*  9884*  9893*  9959*  9979*  10002*  10022*  10097*  10117*
                       10140*  10160*  10171*
PWRMSG 023726           3753   3760#
PWRVEC= 000024           606#  3726*   3727*   3735*  3750*  3751*  4255*  4256*
RDMMR0 025374           4198   4204    4212#
READON 001316           1030#  4682    4695
RESREG= 104414          3417   3717#   3795
RESVEC= 000010           601#
RETRY  001312           1022#  3136*   4079    4081*
SAVREG= 104412          3394   3716#   3780
SDPAR0= 172260           733#  7832*   8224*  10270
SDPAR1= 172262           734#  7833*   8225*
SDPAR2= 172264           735#  8226*
SDPAR3= 172266           736#  8227*
SDPAR4= 172270           737#  8228*   9941*
SDPAR5= 172272           738#  8229*
SDPAR6= 172274           739#  8230*
SDPAR7= 172276           740#  4913    5160    5175   5514   7834*  8231*
SDPDR0= 172220           711#  7819*   8269    8410   8527   8673   8789   8816   8823*  8839*  8859*  8879*  8935
SDPDR1= 172222           712#  7820*
SDPDR2= 172224           713#  7824*
SDPDR3= 172226           714#  7825*
SDPDR4= 172230           715#  7821*   9094*   9233*  9379*  9510*  9640*  9785*  9939*  10074*
SDPDR5= 172232           716#
SDPDR6= 172234           717#
SDPDR7= 172236           718#  5009    5346    5361   5643   7822*  8824*  8840*  8860*  8880*
SIPAR0= 172240           722#  4909    5141    5143   5157   5158   5172   5505   5757   5758*  5761   5766   5770
                        5992   7233*   7314*   7421*  7826*  8216*
SIPAR1= 172242           723#  7234*   7315*   7827*  8217*
SIPAR2= 172244           724#  7235*   7316*   7828*  8218*
SIPAR3= 172246           725#  7236*   7317*   7829*  8219*
SIPAR4= 172250           726#  7239*   7830*   8220*  9100*  9239*  9385*  9516*  9791*
SIPAR5= 172252           727#  8221*
SIPAR6= 172254           728#  8222*
SIPAR7= 172256           729#  6962    7237*   7318*  7422*  7831*  8223*
SIPDR0= 172200           700#  5005    5327    5329   5343   5344   5358   5633   5862   5863*  5866   5871   5875
                        5990   7226*   7325*   7415*  7813*  8273   8313   8387   8414   8421*  8437*  8446   8457*
                        8477*  8531    8579    8669   8713   8812   8848   8931   8983   9776
SIPDR1= 172202           701#  7227*   7319*   7409*  7814*
SIPDR2= 172204           702#  7228*   7320*   7410*  7815*
SIPDR3= 172206           703#  7229*   7321*   7411*  7816*
SIPDR4= 172210           704#  7238*   7817*   7838*  7875*  7878*  9098*  9237*  9383*  9514*  9641*  9789*  9796*
                        9935*  10073*
SIPDR5= 172212           705#
SIPDR6= 172214           706#
SIPDR7= 172216           707#  6965    7230*   7322*  7412*  7818*  8422*  8438*  8458*  8478*
SIZEHI= 177762           631#
SIZELO= 177760           629#  2893    6279    6283   6301   6349   6428   6432   6451
SR0    = 177572          649#
SR1    = 177574          650#
```

```
SR2    = 177576            651#
SR3    = 172516            652#
STACK  = 001100            493#   494    495    496
START    040076           3755   4226   4228   4230   4232   4234   4236   4238   4244#
STKLMT= 177774             501#
STRTAB   001334           1043#  4302
STRT1    040000            891   4225#
STRT2    040010            893   4227#
STRT3    040020            895   4229#
STRT4    040030            897   4231#
STRT5    040040            899   4233#
STRT6    040050            901   4235#
STRT7    040060            903   4237#
STRT8    040070            905   4239#
SUPSTK= 000700             495#  4298   4541   9113   9250   9800
SUPVEC   025334           4200#  7240   7430
SWR    = 177570            503#   504   3106   3138*  3152   3154   3162   3169   3215   3218   3225   3229   3235
                          3242   3290
SW0    = 000001            569#
SW00   = 000001            559#   569
SW01   = 000002            558#   568
SW02   = 000004            557#   567
SW03   = 000010            556#   566
SW04   = 000020            555#   565
SW05   = 000040            554#   564
SW06   = 000100            553#   563
SW07   = 000200            552#   562
SW08   = 000400            551#   561
SW09   = 001000            550#   560
SW1    = 000002            568#
SW10   = 002000            549#
SW11   = 004000            548#
SW12   = 010000            547#
SW13   = 020000            546#
SW14   = 040000            545#
SW15   = 100000            544#
SW2    = 000004            567#
SW3    = 000010            566#
SW4    = 000020            565#
SW5    = 000040            564#
SW6    = 000100            563#
SW7    = 000200            562#   3290
SW8    = 000400            561#
SW9    = 001000            560#   3242
SYSTID= 177764             633#
TAPE1  = ****** U         4220
TAPE2  = ****** U          466   4220
TBIT   = 000020           3821#  3823   3826   3841   4517   7311#  7351
TBITO  = 104416           3718#  7216   8012   8250
TBITOF   024112           3718   3822#
TBITR  = 104420           3719#  7574   8076   9063
TBITRE   024140           3719   3839#
TBITVE= 000014             602#  4261*  4262*
TESTNO   001262           1006#  2837   2839   2842   2844   2847   2849   2851   2853   2856   2858   2860   2862
                          2865   2868   2870   2872   2874   2876   2878   2880   2882   2884   2886   2888   2889
                          2891   2893   2895   2897   2898   2900   2902   2904   2906   2908   2911   2913   2916
```

```
                    2918    2920    2922    2924    2926    2928    2930    2932    2934    2936    2938    2939    2941
                    2944    2945    2947    3204*
TIMEOU  024232      3898#   4875    4907    4939    4971    5003    5035
TKVEC = 000060       609#
TOFLAG  024234      3900#   3924*
TPVEC = 000064       610#
TRAPVE= 000034       608#   4253*   4254*
TRTVEC= 000014       603#
TST1    040536      1043    4296    4325#
TST10   041344      4508    4534#
TST100  062736      8074    8161    8192#
TST101  064162      8194    8381#
TST102  064730      8383    8514#
TST103  065476      8516    8647#
TST104  066260      8649    8782#
TST105  067042      8784    8917#
TST106  067624      8919    9059#
TST107  070426      9061    9204    9226#
TST11   041426      4536    4547    4574#
TST110  071106      9228    9340    9364#
TST111  071506      9366    9477    9503#
TST112  072136      9505    9608    9632#
TST113  072530      9634    9747    9769#
TST114  073174      9771    9895    9920#
TST115  073574      9922   10033   10059#
TST116  074210     10060   10172   10188#
TST12   041632      4576    4637#
TST13   042264      4639    4750#
TST14   042440      4752    4785    4798#
TST15   042604      4800    4865#
TST16   042740      4867    4886    4902#
TST17   043044      4904    4918    4934#
TST2    040660      4330    4342    4359#
TST20   043150      4936    4950    4966#
TST21   043254      4968    4982    4998#
TST22   043360      5000    5014    5030#
TST23   043464      5032    5046    5073#
TST24   043674      5075    5117    5135#
TST25   044104      5137    5179    5197#
TST26   044314      5199    5241    5259#
TST27   044524      5261    5303    5321#
TST3    041014      4361    4401#
TST30   044734      5323    5365    5383#
TST31   045144      5385    5427    5454#
TST32   045274      5456    5479    5497#
TST33   045424      5499    5522    5540#
TST34   045554      5542    5565    5582#
TST35   045710      5584    5608    5625#
TST36   046044      5627    5651    5668#
TST37   046200      5670    5694    5716#
TST4    041060      4403    4425#
TST40   046316      5718    5751#
TST41   046434      5753    5786#
TST42   046552      5788    5821#
TST43   046670      5823    5856#
TST44   047006      5858    5891#
```

```
TST45   047124          5893    5931#
TST46   047226          5933    5972#
TST47   050642          5974    6189#
TST5    041134          4427    4452#
TST50   051436          6191    6292    6341#
TST51   052256          6343    6441    6502#
TST52   052610          6504    6572#
TST53   053002          6574    6625#
TST54   053266          6627    6702#
TST55   053416          6704    6747#
TST56   053662          6749    6810#
TST57   054102          6812    6865#
TST6    041212          4454    4480#
TST60   054300          6867    6914#
TST61   054422          6916    6951#
TST62   054652          6953    7010#
TST63   055050          7012    7059#
TST64   055222          7061    7109#
TST65   055334          7111    7150#
TST66   055576          7152    7212#
TST67   056202          7214    7269    7307#
TST7    041266          4482    4506#
TST70   056566          7309    7404#
TST71   057170          7406    7459    7489#
TST72   057434          7491    7570#
TST73   060230          7572    7727#
TST74   060532          7729    7808#
TST75   061204          7810    7911#
TST76   061656          7913    8008#
TST77   062124          8010    8072#
TYPDS = 104410          3079    3086    3335    3715#
TYPE  = 104400          3073    3080    3087    3220    3228    3307    3309    3314    3316    3344    3367    3373    3414
                        3507    3601    3677    3711#   3752    4291    10258
TYPDC = 104402          3282    3329    3712#
TYPDN = 104406          3714#
TYPDS = 104404          3713#
TYPVAD  022444          3352    3394#
UDPAR0= 177660           689#   7935*   8240*   10268
UDPAR1= 177662           690#   7936*   8241*
UDPAR2= 177664           691#   8242*
UDPAR3= 177666           692#   8243*
UDPAR4= 177670           693#   8244*   10078*
UDPAR5= 177672           694#   8245*
UDPAR6= 177674           695#   8246*
UDPAR7= 177676           696#   4945    5222    5237    5557    7937*   8247*
UDPDR0= 177620           667#   7922*   8261    8402    8543    8665    8808    8924    8951    8958*   8974*   8994*   9014*
UDPDR1= 177622           668#   7923*
UDPDR2= 177624           669#   7927*
UDPDR3= 177626           670#   7928*
UDPDR4= 177630           671#   7924*   9095*   9234*   9380*   9511*   9639*   9786*   9936*   10075*
UDPDR5= 177632           672#
UDPDR6= 177634           673#
UDPDR7= 177636           674#   5041    5408    5423    5686    7925*   8959*   8975*   8995*   9015*
UIPAR0= 177640           678#   4941    5203    5205    5219    5220    5234    5548    5792    5793*   5796    5801    5805
                        5996    7231*   7423*   7929*   8232*
UIPAR1= 177642           679#   7424*   7930*   8233*
```

```
UIPAR2= 177644          680#  7425*  7931*  8234*
UIPAR3= 177646          681#  7426*  7932*  8235*
UIPAR4= 177650          682#  7933*  8236*  9645*  10077*
UIPAR5= 177652          683#  7429*  8237*
UIPAR6= 177654          684#  8238*
UIPAR7= 177656          685#  6963   7232*  7427*  7934*  8239*
UIPDR0= 177600          656#  5037   5389   5391   5405   5406   5420   5676   5897   5898*  5901   5906   5910
                       5994   7225*  7416*  7494*  7916*  8265   8333   8406   8466   8520   8547   8554*  8570*
                       8590*  8599   8610*  8661   8733   8804   8868   8947   9003   10064
UIPDR1= 177602          657#  7219*  7417*  7495*  7917*
UIPDR2= 177604          658#  7220*  7418*  7496*  7918*
UIPDR3= 177606          659#  7221*  7419*  7497*  7919*
UIPDR4= 177610          660#  7498*  7920*  7941*  7975*  7978*  9096*  9235*  9381*  9512*  9643*  9787*  9937*
                      10084*
UIPDR5= 177612          661#  7428*  7499*
UIPDR6= 177614          662#  7500*
UIPDR7= 177616          663#  6966   7222*  7420*  7501*  7921*  8555*  8571*  8591*  8611*
USESTK= 000600          496#  4300   4543   9656
USEVEC  025356         4206#  7242   7432
WBIT  = 000100         8095#  8104   8111   8148   8156
WORK    074764        10232*  10233  10236  10237  10238  10240*  10241  10261*  10274#
$BDADR  001122          954#
$BDDAT  001126          956#
$BELL   001210          982#  3220   3252
$CHARC  023052         3523   3532*  3537#
$CLR.T  021220         3092   3095#
$CMTAG  001100          942#  4244
$CM1  = 000006          968#   969#   970#   971#   972#   973#   974#
$CM2  = 000014          968#   969#   970#   971#   972#   973#   974#
$CM3  = 000006          966#   968
$CM4  = 000006          974#   975#   976#   977#   978#   979#   980#
$CRLF   001215          984#  3087   3228   3252   3309   3316   3507   3540
$DBLK   023520         3643   3677   3685#
$DB20   023772         3341   3364   3411   3780#
$DOAGN  021240         3069   3090   3097   3103#
$DTBL   023510         3646   3681#
$ENDAD  021230          929   3099#  3232   4289
$ENDCT  021056         3071#  4257
$ENULL  021306         3118#
$EOP    021022         3061#  10194  10281
$EOPCT  021050         3068#  3072   4257*
$ERFLG  001103          945#  3126   3158   3160   3166*  3187   3212*  3252   3919   3949   3983
$ERMAX  001115          951#  3160   3182*  3187   4260*
$ERROR  021576         2928   3203#  4251
$ERRPC  001116          952#  2847   2849   2851   2853   2856   2860   2868   2870   2872   2874   2876   2878
                       2880   2882   2884   2886   2888   2889   2891   2893   2895   2897   2898   2900   2902
                       2904   2906   2908   2911   2913   2916   2918   2920   2922   2924   2926   2930   2932
                       2934   2936   2938   2939   2941   3222*  3223*  3224   3252   3281
$ERRTB  001356         1068#  3300   3304
$ERTTL  001112          949#  3084   3088*  3221*  3252
$ESCAP  001206          981#  3181*  3238   3240   3252   4259*
$FILLC  001150          964#  3510   3540
$FILLS  001147          963#  3540
$GDADR  001120          953#
$GDDAT  001124          955#
$GET42  021202         3089#
```

```
$HD    = 000000          475
$ICNT    001104          946# 3173* 3174  3176* 3186
$ILLUP   023720         3726  3757#
$ITEMB   001114          950# 3224* 3252  3279
$LF      001216          985# 3252  3540
$LOOP    021300         3111  3115#
$LPADR   001106          947# 3164* 3179* 3184  3186  4082  4265* 4326* 4579* 4870* 5982* 6321  6471
                        6508* 7577* 8079* 8248* 9066* 9089* 9374* 9779* 9930* 10067*
$LPERR   001110          948# 3164  3180* 3186  3237  4266* 4327* 4332* 4340* 4363* 4370* 4377* 4383*
                        4390* 4405* 4412* 4429* 4439* 4456* 4468* 4484* 4494* 4510* 4522* 4580* 4590*
                        4597* 4603* 4617* 4643* 4652* 4658* 4669* 4680* 4692* 4706* 4720* 4735* 4756*
                        4764* 4771* 4779* 4809* 4821* 4829* 4835* 4871* 4876* 4883* 4908* 4915* 4940*
                        4947* 4972* 4979* 5004* 5011* 5036* 5043* 5078* 5094* 5115* 5140* 5156* 5177*
                        5202* 5218* 5239* 5264* 5280* 5301* 5326* 5342* 5363* 5388* 5404* 5425* 5460*
                        5477* 5503* 5520* 5546* 5563* 5588* 5606* 5631* 5649* 5674* 5692* 5720* 5730*
                        5739* 5755* 5765* 5774* 5790* 5800* 5809* 5825* 5835* 5844* 5860* 5870* 5879*
                        5895* 5905* 5914* 5935* 5943* 5951* 5983* 6008* 6022* 6036* 6050* 6064* 6078*
                        6092* 6106* 6120* 6134* 6148* 6162* 6176* 6200* 6205* 6213* 6246* 6269* 6278*
                        6291* 6359* 6367* 6399* 6418* 6427* 6440* 6509* 6527* 6559* 6580* 6612* 6637*
                        6649* 6674* 6688* 6716* 6725* 6758* 6767* 6780* 6795* 6824* 6838* 6852* 6880*
                        6887* 6901* 6927* 6935* 6993* 7014* 7045* 7065* 7078* 7092* 7114* 7135* 7158*
                        7197* 7244* 7268* 7323* 7380* 7434* 7458* 7502* 7536* 7578* 7608* 7649* 7659*
                        7674* 7687* 7704* 7713* 7748* 7764* 7781* 7792* 7839* 7847* 7852* 7860* 7865*
                        7874* 7883* 7889* 7896* 7942* 7947* 7952* 7960* 7965* 7974* 7983* 7989* 7996*
                        8013* 8048* 8080* 8100* 8107* 8114* 8121* 8129* 8137* 8144* 8151* 8159* 8249*
                        8252* 8363* 8385* 8496* 8518* 8629* 8651* 8764* 8786* 8899* 8921* 9034* 9067*
                        9090* 9105* 9130* 9150* 9173* 9193* 9203* 9260* 9282* 9302* 9325* 9338* 9375*
                        9402* 9422* 9445* 9465* 9475* 9527* 9549* 9569* 9592* 9605* 9673* 9693* 9716*
                        9736* 9746* 9780* 9820* 9840* 9863* 9883* 9894* 9931* 9958* 9978* 10001* 10021*
                       10031* 10068* 10096* 10116* 10139* 10159* 10169*
$MXCNT   021574         3177  3186#
$NULL    001146          962# 3512  3540
$NWTST=  000001         4314# 4316  4350# 4352  4393# 4395  4416# 4418  4442# 4444  4471# 4473  4498#
                        4500  4525# 4527  4562# 4564  4620# 4622  4738# 4740  4788# 4790  4853# 4855
                        4890# 4892  4922# 4924  4954# 4956  4986# 4988  5018# 5020  5059# 5061  5121#
                        5123  5183# 5185  5245# 5247  5307# 5309  5369# 5371  5440# 5442  5483# 5485
                        5526# 5528  5569# 5571  5612# 5614  5655# 5657  5707# 5709  5742# 5744  5777#
                        5779  5812# 5814  5847# 5849  5882# 5884  5918# 5920  5963# 5965  6178# 6180
                        6330# 6332  6491# 6493  6561# 6563  6615# 6617  6692# 6694  6728# 6730  6798#
                        6800  6855# 6857  6904# 6906  6938# 6940  6996# 6998  7048# 7050  7096# 7098
                        7138# 7140  7200# 7202  7290# 7292  7385# 7387  7479# 7481  7559# 7561  7716#
                        7718  7796# 7798  7899# 7901  7999# 8001  8062# 8064  8177# 8179  8366# 8368
                        8499# 8501  8632# 8634  8767# 8769  8902# 8904  9047# 9049  9214# 9216  9350#
                        9352  9489# 9491  9619# 9621  9756# 9758  9906# 9908  10043# 10045 10183# 10185
$OCNT    023300         3573* 3602* 3615#
$OCTVL   024074         3782  3807#
$OMODE   023302         3568* 3572* 3577  3580* 3591* 3617#
$OVER    021560         3139  3157  3165  3175  3183#
$PASS    001100          943# 3065* 3066* 3077  3117  3171  3187
$PWRAD   023714         3755#
$PWRDN   023572         3726# 3750  4255
$PWRMG   023710         3753#
$PWRUP   023640         3735  3740#
$QUES    001214          983# 3252
$RDCHR=  ******  U      3716
$RDDEC=  ******  U      3716
```

```
$RDLIN= ****** U        3716
$RDOCT= ****** U        3716
$REGAD  001152           966#
$REG0   001154           968#  2860  2868  2870  2872  2874  2876  2878  2880  2886  2891  2898  2926
                        2930  2936  2944  2945  2947  3206*
$REG1   001156           969#  2847  2849  2855  2868  2872  2874  2878  2880  2889  2895  2898  2908
                        2911  2918  2924  2928  2930  2934  2945  2947  3207*
$REG2   001160           970#  2849  2851  2853  2856  2860  2868  2874  2878  2880  2886  2920  2928
                        2947  3208*
$REG3   001162           971#  2853  2886  2895  2924  2926  3209*
$REG4   001164           972#  2947  3210*
$REG5   001166           973#  3211*
$RESRE  022610          3456#  3717
$RTRN   021276          3093   3112#  4261  4263*
$SAVRE  022552          3440#  3716
$SAVR6  023724          3734*  3740  3741*  3742*  3759#
$SCOPE  021312          3135#  4249
$SETUP= 000037           880#  3063  3229  4289
$STUP = 177777           880#
$SVLAD  021532          3147   3178#
$SVPC = 000240           927#   932
$SWR  = 177400           466#   475   480   481   482   483   484   485   486   487   980   981   982
                        3057   3064  3091  3104  3117  3127  3128  3129  3130  3131  3138  3150  3152
                        3153   3158  3159  3160  3167  3168  3169  3180  3183  3186  3194  3195  3196
                        3197   3198  3199  3215  3218  3225  3229  3235  3252  4330  4363  4405  4429
                        4456   4484  4510  4538  4578  4641  4754  4802  4869  4906  4938  4970  5002
                        5034   5077  5139  5201  5263  5325  5387  5458  5501  5544  5586  5629  5672
                        5720   5755  5790  5825  5860  5895  5935  5976  6193  6345  6506  6576  6629
                        6706   6751  6814  6869  6918  6955  7014  7063  7113  7154  7216  7311  7408
                        7493   7574  7731  7812  7915  8012  8076  8196  8385  8518  8651  8786  8921
                        9063   9230  9368  9507  9636  9773  9924  10060 10189
$SWRMK= 000200           466#   487   488  3131  3132  3154  3155
$TBIT   021304          3108*  3117#  4264*
$TIMES  001204           980#  3064*  3167*  3174  3177*  3186  4258*  5458*  5501*  5544*  5586*  5629*  5672*
                        6506*  6576*  7493*
$TKB    001140           959#
$TKS    001136           958#  4246
$TMP0   001170           974#  2902  2913  2922  2941  4708*  4718  4722*  4732  5101*  5108*  5163*  5170*
                        5225*  5232*  5287*  5294*  5349*  5356*  5411*  5418*  6009*  6021  6023*  6035  6037*
                        6049   6051*  6063  6065*  6077  6079*  6091  6093*  6105  6107*  6119  6121*  6133
                        6135*  6147  6149*  6161  6163*  6175  6207*  6222  6361*  6376  6400*  6404  7031*
                        7180*  7189  8103*  8104  8110*  8111  8117*  8118  8125*  8126  8133*  8134  8140*
                        8141   8147*  8148  8155*  8156  8259*  8309*  8329*  8349*  8352  8392*  8442*  8462*
                        8482*  8485  8525*  8575*  8595*  8615*  8618  8659*  8709*  8729*  8749*  8752  8794*
                        8844*  8864*  8884*  8887  8929*  8979*  8999*  9019*  9022  10254*
$TMP1   001172           975#  2882  2884  2913  2941  6316*  6466*  7181*  7186  10255*
$TMP2   001174           976#  2882  2913  2941  6212*  6366*  7182*  7183  10253*
$TMP3   001176           977#  7179*  7185*  7188*  7191*  7192  7352*  7357*  7364*  7368*  7371*  7372
$TMP4   001200           978#
$TMP5   001202           979#  4887*  4919*  4951*  4983*  5015*  5047*  5118*  5180*  5242*  5304*  5366*  5428*
                        5480*  5523*  5566*  5609*  5652*  5695*
$TN   = 000117           466#   475  4296  4314  4330#  4342  4350  4360  4363#  4393  4402  4405#  4416
                        4426   4429#  4442  4453  4456#  4471  4481  4484#  4498  4507  4510#  4525  4535
                        4538#  4547  4562  4575  4578#  4620  4638  4641#  4738  4751  4754#  4785  4788
                        4799   4802#  4853  4866  4869#  4886  4890  4903  4906#  4918  4922  4935  4938#
                        4950   4954  4967  4970#  4982  4986  4999  5002#  5014  5018  5031  5034#  5046
```

```
                    5059    5074    5077#   5117    5121    5136    5139#   5179    5183    5198    5201#   5241    5245
                    5260    5263#   5303    5307    5322    5325#   5365    5369    5384    5387#   5427    5440    5455
                    5458#   5479    5483    5498    5501#   5522    5526    5541    5544#   5565    5569    5583    5586#
                    5608    5612    5626    5629#   5651    5655    5669    5672#   5694    5707    5717    5720#   5742
                    5752    5755#   5777    5787    5790#   5812    5822    5825#   5847    5857    5860#   5882    5892
                    5895#   5918    5932    5935#   5963    5973    5976#   5981    6178    6190    6193#   6292    6330
                    6342    6345#   6441    6491    6503    6506#   6507    6561    6573    6576#   6615    6626    6629#
                    6692    6703    6706#   6728    6748    6751#   6798    6811    6814#   6855    6866    6869#   6904
                    6915    6918#   6938    6952    6955#   6996    7011    7014#   7048    7060    7063#   7096    7110
                    7113#   7138    7151    7154#   7200    7213    7216#   7269    7290    7308    7311#   7385    7405
                    7408#   7459    7479    7490    7493#   7559    7571    7574#   7576    7716    7728    7731#   7796
                    7809    7812#   7899    7912    7915#   7999    8009    8012#   8062    8073    8076#   8078    8161
                    8177    8193    8196#   8366    8382    8385#   8499    8515    8518#   8632    8648    8651#   8767
                    8783    8786#   8902    8918    8921#   9047    9060    9063#   9065    9204    9214    9227    9230#
                    9340    9350    9365    9368#   9477    9489    9504    9507#   9608    9619    9633    9636#   9747
                    9756    9770    9773#   9895    9906    9921    9924#   10033   10043   10060#  10172   10183   10189#
$TPB        001144   961#   3529*   3540
$TPFLG      001151   965#   3490    3540
$TPS        001142   960#   3527    3540
$TRAP       023530  3695#   4253
$TRP    =   000022  3702#   3712#   3713#   3714#   3715#   3716#   3717#   3718#   3719#   3720#
$TRPAD      023550  3699    3710#
$TSTNM      001102   944#   3063*   3126    3156    3178*   3183    3187    3204    3214    3252    4328*   4329    4581*
                    4582    4872*   4873    5984*   5985    6510*   6511    7575*   7580    8081*   8082    9068*   9069
$TYPBN=  ****** U   3716
$TYPDS      023304  3631#   3715
$TYPE       022646  3490#   3702    3711
$TYPEC      023006  3509    3516    3522    3527#   3528
$TYPEX      023054  3533    3535    3538#
$TYPOC      023102  3571#   3712
$TYPON      023116  3570    3573#   3714
$TYPOS      023056  3566#   3713
$XTSTR      021330  3141#
$$GET4=     000001  3091#   3096
$$TRP   =   000002  3701#   3712    3713    3714    3715    3716    3717    3718    3719    3720
$OFILL      023301  3567*   3571*   3581    3616#
.       =   074776   883#    887     889#    927     928#    930#    932#    940#    986    3076#   3117    3119#   3186
                    3187    3252    3381#   3540    3685#   3737    3758    3807#   4223#   4265    4266    7691*   7697
```

```
ADD18    466#  6008  6022  6036  6050  6064  6078  6092  6106  6120  6134  6148  6162
AND      466#  3916  3942  3946  3972  3976  3980
BYTADR   466#  5720  5755  5790  5825  5860  5895
BYTEMG  5742#  5744  5779  5814  5849  5884
CMPPDR   466#  8292  8312  8332  8425  8445  8465  8558  8578  8598  8692  8712  8732  8827  8847
        8867  8962  8982  9002
COMMEN     1#   879#
COUNT1   466#  5459  5502  5545  5587  5630  5673
DUAL     466#  5077  5139  5201  5263  5325  5387
DUALMA   466#  8252  8385  8518  8651  8786  8921
DUALMG  5121#  5123  5185  5247  5309  5371
ENDCOM     1#   879#
ENTRY    466#  4325  4578  4869  5981  6507  7576  8078  9065
ERROR    497#  3913  3921  3929  3951  3953  3985  3987  4023  4025  4069  4078  4123  4127  4167
        4169  4197  4203  4209  4346  4369  4375  4382  4388  4411  4437  4466  4493  4521  4548
        4589  4596  4602  4614  4651  4657  4667  4678  4690  4704  4717  4731  4763  4770  4778
        4786  4807  4817  4827  4834  4888  4920  4952  4984  5016  5048  5086  5119  5148  5181
        5210  5243  5272  5305  5334  5367  5396  5429  5481  5524  5567  5610  5653  5696  5729
        5738  5764  5773  5799  5808  5834  5843  5869  5878  5904  5913  5949  6020  6034  6048
        6062  6076  6090  6104  6118  6132  6146  6160  6174  6214  6221  6230  6257  6277  6289
        6306  6320  6368  6375  6384  6407  6426  6438  6456  6470  6545  6549  6598  6602  6648
        6661  6687  6724  6766  6774  6779  6791  6794  6834  6837  6848  6851  6886  6897  6900
        6934  6978  6981  6990  7032  7039  7077  7090  7117  7132  7194  7249  7282  7374  7439
        7472  7523  7531  8039  8043  8106  8113  8120  8128  8136  8143  8150  8158  8299  8319
        8339  8355  8432  8452  8472  8488  8565  8585  8605  8621  8699  8719  8739  8755  8834
        8854  8874  8890  8969  8989  9009  9025  9116  9118  9138  9158  9180  9201  9210  9248
        9271  9293  9314  9337  9346  9410  9430  9452  9473  9483  9538  9560  9581  9604  9614
        9659  9661  9681  9701  9723  9744  9753  9806  9808  9828  9848  9870  9891  9901  9966
        9986 10008 10029 10039 10104 10124 10146 10167 10178 10257
ESCAPE     1#   879#
LOADPD   466#  5999  8260  8264  8268  8272  8276  8280  8393  8397  8401  8405  8409  8413  8526
        8530  8534  8538  8542  8546  8660  8664  8668  8672  8676  8680  8795  8799  8803  8807
        8811  8815  8930  8934  8938  8942  8946  8950  9085  9370  9775  9926 10063
MGDUAL  8177#  8179  8368  8501  8634  8769  8904
MMLOAD   466#  6512  7581  8083  9070
MSG1    4314#  4316
MSG10   4525#  4527
MSG100  8177#  8179
MSG101  8366#  8368
MSG102  8499#  8501
MSG103  8632#  8634
MSG104  8767#  8769
MSG105  8902#  8904
MSG106  9047#  9049
MSG107  9214#  9216
MSG11   4562#  4564
MSG110  9350#  9352
MSG111  9489#  9491
MSG112  9619#  9621
MSG113  9756#  9758
MSG114  9906#  9908
MSG115 10043# 10045
MSG116 10183# 10185
MSG12   4619#  4622
MSG13   4738#  4740
MSG14   4788#  4790
```

```
MSG15     4853#   4855
MSG16     4890#   4892
MSG17     4922#   4924
MSG2      4350#   4352
MSG20     4954#   4956
MSG21     4986#   4988
MSG22     5018#   5020
MSG23     5059#   5061
MSG24     5121#   5123
MSG25     5183#   5185
MSG26     5245#   5247
MSG27     5307#   5309
MSG3      4393#   4395
MSG30     5369#   5371
MSG31     5440#   5442
MSG32     5483#   5485
MSG33     5526#   5528
MSG34     5569#   5571
MSG35     5612#   5614
MSG36     5655#   5657
MSG37     5707#   5709
MSG4      4416#   4418
MSG40     5742#   5744
MSG41     5777#   5779
MSG42     5812#   5814
MSG43     5847#   5849
MSG44     5882#   5884
MSG45     5918#   5920
MSG46     5963#   5965
MSG47     6178#   6180
MSG5      4442#   4444
MSG50     6330#   6332
MSG512    6491#   6493    6563
MSG53     6615#   6617
MSG54     6692#   6694
MSG55     6728#   6730
MSG56     6798#   6800
MSG57     6855#   6857
MSG6      4471#   4473
MSG60     6904#   6906
MSG61     6938#   6940
MSG62     6996#   6998
MSG63     7048#   7050
MSG64     7096#   7098
MSG65     7138#   7140
MSG66     7200#   7202
MSG67     7290#   7292
MSG7      4498#   4500
MSG70     7385#   7387
MSG71     7479#   7481
MSG72     7559#   7561
MSG73     7716#   7718
MSG74     7796#   7798
MSG75     7899#   7901
MSG76     7999#   8001
MSG77     8062#   8064
```

```
MULT        1#    879#
NEWTST      1#    879#  4314  4350  4393  4416  4442  4471  4498  4525  4562  4620  4738  4788  4853
         4890  4922  4954  4986  5018  5059  5121  5183  5245  5307  5369  5440  5483  5526  5569
         5612  5655  5707  5742  5777  5812  5847  5882  5918  5963  6178  6330  6491  6561  6615
         6692  6728  6798  6855  6904  6938  6996  7048  7096  7138  7200  7290  7385  7479  7559
         7716  7796  7899  7999  8062  8177  8366  8499  8632  8767  8902  9047  9214  9350  9489
         9619  9756  9906 10043 10183
PAGEF     466#  6524  6577
PARMSG   5483#  5485  5528
PDRMSG   5612#  5614  5657
POP         1#   879#  3461  3672  3744
PUSH        1#   879#  3441  3631  3728
RESPMG   4890#  4892  4924  4956  4988  5020
RESPON    466#  4874  4906  4938  4970  5002  5034
SAVTST    466#  3204
SCOPE     498#  4360  4402  4426  4453  4481  4507  4535  4575  4638  4751  4799  4866  4903  4935
         4967  4999  5031  5074  5136  5198  5260  5322  5384  5455  5498  5541  5583  5626  5669
         5717  5752  5787  5822  5857  5892  5932  5973  6190  6342  6503  6573  6626  6703  6748
         6811  6866  6915  6952  7011  7060  7110  7151  7213  7308  7405  7490  7571  7728  7809
         7912  8009  8073  8193  8382  8515  8648  8783  8918  9060  9227  9365  9504  9633  9770
         9921 10059 10188 10278
SETACF    466#  6706  6751  6814  6869  6918
SETTRA   3702#  3712  3713  3714  3715  3716  3717  3718  3719
SETUP       1#   879#
SKIP        1#   879#  4296  4342  4547  4785  4886  4918  4950  4982  5014  5046  5117  5179  5241
         5303  5365  5427  5479  5522  5565  5608  5651  5694  6292  6441  7269  7459  8161  9204
         9340  9477  9608  9747  9895 10033 10172
SLASH       1#   879#
SPACE     879#
SSCOPE    466#  3136
STARS       1#   879#   907   933  1052  3049  3120  3187  3422  3468  3540  3618  3686  3721  3768
         4314  4324  4350  4358  4393  4400  4416  4424  4442  4451  4471  4479  4498  4505  4525
         4533  4562  4573  4620  4636  4738  4749  4788  4797  4853  4864  4890  4901  4922  4933
         4954  4965  4986  4997  5018  5029  5059  5072  5121  5134  5183  5196  5245  5258  5307
         5320  5369  5382  5440  5453  5483  5496  5526  5539  5569  5581  5612  5624  5655  5667
         5707  5715  5742  5750  5777  5785  5812  5820  5847  5855  5882  5890  5918  5930  5963
         5971  6178  6188  6330  6340  6491  6501  6561  6571  6615  6624  6692  6701  6728  6746
         6798  6809  6855  6864  6904  6913  6938  6950  6996  7009  7048  7058  7096  7108  7138
         7149  7200  7211  7290  7306  7385  7403  7479  7488  7559  7569  7716  7726  7796  7807
         7899  7910  7999  8007  8062  8071  8177  8191  8366  8380  8499  8513  8632  8646  8767
         8781  8902  8916  9047  9058  9214  9225  9350  9363  9489  9502  9619  9631  9756  9768
         9906  9919 10043 10058 10183 10187
TRACK     466#  4359  4401  4425  4452  4480  4506  4534  4574  4637  4750  4798  4865  4902  4934
         4966  4998  5030  5073  5135  5197  5259  5321  5383  5454  5497  5540  5582  5625  5668
         5716  5751  5786  5821  5856  5891  5931  5972  6189  6341  6502  6572  6625  6702  6747
         6810  6865  6914  6951  7010  7059  7109  7150  7212  7307  7404  7489  7570  7727  7808
         7911  8008  8072  8192  8381  8514  8647  8782  8917  9059  9226  9364  9503  9632  9769
         9920
TRACK1    466#  4360  4402  4426  4453  4481  4507  4535  4575  4638  4751  4799  4866  4903  4935
         4967  4999  5031  5074  5136  5198  5260  5322  5384  5455  5498  5541  5583  5626  5669
         5717  5752  5787  5822  5857  5892  5932  5973  6190  6342  6503  6573  6626  6703  6748
         6811  6866  6915  6952  7011  7060  7110  7151  7213  7308  7405  7490  7571  7728  7809
         7912  8009  8073  8193  8382  8515  8648  8783  8918  9060  9227  9365  9504  9633  9770
         9921
TRMTRP    3702#
TYPBIN      1#   879#
```

```
TYPDEC      1#    879#  3077  3084
TYPNAM      1#    879#  4287
TYPNUM      1#    879#
TYPOCS      1#    879#
TYPOCT      1#    879#
TYPTXT      1#    879#  3073  3080
USER      466#    986
ZEROER    466#   3242
$$CMRE    933#    968   969   970   971   972   973
$$CMTM    933#    974   975   976   977   978   979
$$ESCA      1#    879#
$$NEWT      1#    879#  4314  4350  4393  4416  4442  4471  4498  4525  4562  4620  4738  4788  4853
         4890   4922  4954  4986  5018  5059  5121  5183  5245  5307  5369  5440  5483  5526  5569
         5612   5655  5707  5742  5777  5812  5847  5882  5918  5963  6178  6330  6491  6561  6615
         6692   6728  6798  6855  6904  6938  6996  7048  7096  7138  7200  7290  7385  7479  7559
         7716   7796  7899  7999  8062  8177  8366  8499  8632  8767  8902  9047  9214  9350  9489
         9619   9756  9906 10043 10183
$$SET     3702#  3712  3713  3714  3715  3716  3717  3718  3719
$$SKIP      1#   879#  4296  4342  4547  4785  4886  4918  4950  4982  5014  5046  5117  5179  5241
         5303   5365  5427  5479  5522  5565  5608  5651  5694  6292  6441  7269  7459  8161  9204
         9340   9477  9608  9747  9895 10033 10172
.EQUAT      1#
.HEADE      1#    466#
.KT11       1#
.SETUP      1#    466#   880
.SWRHI      1#    466#   475
.SWRLO    488#
.$ACT1      1#    466#   907
.$CATC      1#    466#   880
.$CMTA      1#    466#   933
.$DB2D      1#
.$DB2O      1#    466#  3768
.$DIV       1#
.$EOP       1#    466#  3049
.$ERRO      1#    466#  3187
.$ERRT      1#
.$MULT      1#
.$POWE      1#    466#  3721
.$RAND      1#
.$RDDE      1#
.$RDOC      1#
.$READ      1#
.$SAVE      1#    466#  3422
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#    466#  3120
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#    466#  3686
.$TYPB      1#
.$TYPD      1#    466#  3618
.$TYPE      1#    466#  3468
.$TYPO      1#    466#  3540
.1170       1#    466#   489
```

. ABS. 074776 000

ERRORS DETECTED: 0

DSKZ:CEKBEC.BIN,CEKBEC.LST/CRF/SOL/NL:TOC=DSKZ:CEKBEC.SML,CEKBEC.P11
RUN-TIME: 26 39 3 SECONDS
RUN-TIME RATIO: 294/69=4.2
CORE USED: 39K (77 PAGES)