KD11-E

11/34 MEMORY MANAGEMENT

MD-11-DFKTH-A

EP-DFKTH-A-DL-A

COPYRIGHT © 1977

FICHE 1 OF 1

APR 1977

MADE IN USA

.REM @

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DFKTH-A-D
PRODUCT NAME: PDP 11/34 MEMORY MANAGEMENT DIAGNOSTIC
DATE: JANUARY 31,1977
MAINTAINER: DIAGNOSTIC PROGRAMMIMG
AUTHOR: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1977 BY DIGITAL EQUIPMENT CORPORATION

```
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
```

PROGRAM HISTORY

DATE	REVISION	REASON FOR REVISION
31-JAN-77	A	FIRST RELEASE

TABLE OF CONTENTS

1.0 PROGRAM INFORMATION

1.1 ABSTRACT

THIS PROGRAM WAS DESIGNED USING A "BOTTOM UP" APPROACH STARTING WITH THE SMALLEST SEGMENT OF MEMORY MANAGMENT LOGIC POSSIBLE AND BUILDING TO COVER ALL OF THE LOGIC. THE DIAGNOSTIC WILL PROVIDE ENOUGH INFORMATION SUCH THAT BY DEDUCTION, THE FAILURE CAN BE ISOLATED TO A SMALL SEGMENT OF THE MEMORY MANAGEMENT LOGIC.

THE PROGRAM BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC, THEN WORKS OUTWARD THROUGH THE MEMORY MANAGEMENT REGISTERS. AFTER THE REGISTERS ARE FOUND TO BE USEABLE, RELOCATION (CONSTRUCTION OF PHYSICAL ADDRESSES FROM A VIRTUAL ADDRESS AND THE ASSOCIATED PAR/PDR INFORMATION) IS TESTED FOLLOWED BY TESTING OF THE ABORT AND STATUS SEGMENTS OF LOGIC. FINALLY, CHECKS OF SPECIAL ABORT SEQUENCES AND TESTING OF THE MFPI/MTPI INSTRUCTIONS ARE DONE.

1.2 REQUIREMENTS

A PDP 11/34 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY.

1.3 RELATED DOCUMENTS AND STANDARDS

1. ACT11/XXDP PROGRAMMING SPECIFICATION
2. STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE
3. DIAGNOSITC ENGINEERING STANDARDS AND CONVENTIONS
4. PDP11 MAINDEC SYSMAC PACKAGE
5. XXDP USER'S MANUAL

1.4 PRELIMINARY PROGRAMS

BEFORE THIS MEMORY MANAGEMENT DIAGNOSITC IS RUN, THE FOLLOWING CPU DIAGNOSTICS SHOULD BE RUN:

MD-11-DFKAA	PDP 11/34 BASIC CPU TESTS
MD-11-DFKAB	PDP 11/34 TRAPS TESTS

ALSO, ONE OF THE MAIN MEMORY DIAGNOSTICS SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURES

THE PROGRAM IS SUPPLIED ON THE DIAGNOSTIC LOAD MEDIA.
REFER TO THE XXDP USER'S MANUAL FOR FURTHER INFORMATION.
FOR USE WITH ACT OR APT, REFER TO THEIR RESPECTIVE
DOCUMENTS. THE PROGRAM CAN ALSO BE DIRECTLY LOADED
USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

2.2 STARTING PROCEDURES

THE PROGRAM IS STARTED BY LOADING ADDRESS 200 AND
STARTING. IF THE PROCESSOR HAS THE OPTIONAL PROGRAMMER'S
CONSOLE, THE SWITCH REGISTER SHOULD BE SET ACCORDING TO
SECTION 2.3 BEFORE THE PROGRAM IS STARTED. IF THERE
IS NO HARDWARE SWITCH REGISTER, THE PROGRAM WILL USE THE
SOFTWARE SWITCH REGISTER AT LOCATION 176 (LOCATION 174
WILL BE USED AS THE SOFTWARE DISPLAY REGISTER). IN THAT CASE
THE PROGRAM WILL ASK FOR THE INITIAL SWITCH REGISTER
VALUE BY TYPING "SWR= XXXXXX NEW= " AFTER TYPING
THE NAME OF THE PROGRAM (XXXXXX = THE OCTAL CONTENTS OF
LOCATION 176). (SEE SECTION 2.4)

ALSO THE PROGRAM CAN BE MADE TO USE THE SOFTWARE SWITCH
REG. EVEN IF THE HARDWARE SWITCH REG. IS PRESENT BY LOADING
"177777" INTO THE HARDWARE SWITCH REG. BEFORE STARTING
THE PROGRAM.

2.3 CONTROL SWITCH SETTINGS

SWITCH	OCTAL VALUE	USE
SW15	100000	HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING).
SW14	040000	LOOP ON TEST THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH

```
                                                    NORMALLY TAKES PLACE DURING
                                                    EVERY OTHER PASS STARTING
                                                    WITH  THE THIRD PASS.

                        SW11    004000          INHIBIT SUBTEST ITERATIONS
                                                    THIS SWITCH WHEN SET INHIBITS
                                                    ITERATIONS OF EACH SUBTEST AFTER
                                                    THE FIRST PASS.  IF THIS SWITCH
                                                    IS NOT SET, EACH SUBTEST IS RUN
                                                    200. TIMES.

                        SW10    002000          BELL ON ERROR
                                                    THIS SWITCH WHEN SET WILL RING
                                                    THE CONSOLE TERMINAL BELL WHEN
                                                    AN ERROR HAS BEEN DETECTED.

                        SW9     001000          LOOP ON ERROR
                                                    THIS SWITCH WHEN SET WILL
                                                    CAUSE THE PROGRAM TO LOOP ON THE
                                                    FIRST FAILURE WHICH IS ENCOUNTERED
                                                    EVEN IF THE FAILURE IS INTERMITTANT

                        SW8     000400          LOOP ON TEST IN SWR<7:0>
                                                    THIS SWITCH WHEN SET WILL
                                                    CAUSE THE PROGRAM TO LOOP ON THE
                                                    TEST WHOSE TEST NUMBER IS SET
                                                    IN BITS 7-0 OF THE SWITCH REG.


2.4     LOADING THE SWITCH REGISTER
        ---------------------------

        THE HARDWARE SWITCH REGISTER PROVIDED WHEN THE OPTIONAL
        PROGRAMMER'S CONSOLE IS PRESENT IS LOADED DIRECTLY FROM
        THE CONSOLE KEYPAD BY DEPRESSING THE "LSR" KEY.  THE
        VALUE OF THE HARDWARE SWITCH REG. CAN BE CHANGED ANY
        TIME WHETHER THE PROGRAM IS RUNNING OR NOT.

        TO LOAD THE SOFTWARE SWITCH REG. WHILE THE PROGRAM IS
        RUNNING, A CONTROL G (↑G) SHOULD BE TYPED ON THE CONSOLE
        TERMINAL. (THE "SCOPE" AND "ERROR" ROUTINES CHECK TO SEE
        IF A ↑G HAS BEEN TYPED.)  THE ORIGINAL VALUE OF THE SOFTWARE
        SWTICH REG. WILL BE REQUESTED AS MENTIONED IN SECTION 2.2.

        IN RESPONSE TO A ↑G OR AT THE BEGINNING OF THE PROGRAM, THE
        PROGRAM WILL TYPE:

                SWR = XXXXXX  NEW =

        WHERE "XXXXXX" IS THE CURRENT OCTAL CONTENTS OF LOC. 176.
        THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING:
            XXXXXX<CR>  ONE TO SIX OCTAL DIGITS FOLLOWED BY A
                        CARRIAGE RETURN WHICH WILL BE LOADED
                        AS THE NEW VALUE FOR THE SWITCH REG.
                <CR>    JUST A <CR>, LEAVES THE SWITCH REG.
                        AS IT IS.
                XXX↑U   A CONTROL-U (↑U) WILL CAUSE ALL OF THE
```

DIGITS TYPED SO FAR TO BE IGNORED.

↑C — WILL CAUSE THE PROGRAM TO TYPE THE PRESENT TEST AND PASS NUMBERS, REQUEST A NEW VALUE FOR THE SWITCH REG., AND JUMP TO THE END-OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY TO THE NEXT PASS WITH A NEW SW. REG. VALUE

<ILL.CHAR> — ANY CHARACTER TYPED WHICH IS NOT ANY OF THE ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM TO TYPE A "?<CRLF>" AND REACT AS THOUGH A ↑U HAD BEEN TYPED.

NOTE: RECOGNITION OF A ↑G MAY BE HAMPERED BY EXECUTION OF A COUPLE "RESET" INSTRUCTIONS WITHIN THE PROGRAM.

2.5 EXECUTION TIMES

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS OR TRACE TRAPPING IS APPROXIMATELY 5 SECONDS.

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS AND TRACE TRAPPING ENABLED IS APPROXIMATELY 3 1/4 MINUTES.

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE ERROR HANDLING ROUTINE ($ERROR). THE VALUE OF BITS 15,13,10, AND 9 IN THE SWITCH REGISTER ARE CONSIDERED IN REPORTING AN ERROR (SEE SECTION 2.3). THE ERROR INFORMATION WILL BE TYPED UNLESS SW13 = 1.

IF SW15 = 1, THE PROCESSOR WILL HALT AFTER THE ERROR IS REPORTED. IF THE CONTENTS OF THE SOFTWARE SWITCH REGISTER ARE TO BE CHANGED, A ↑G SHOULD BE TYPED BEFORE PRESSING "CONTINUE" TO RESUME TESTING.

IF SW9 = 1 (LOOP ON ERROR), THE PROGRAM WILL GO TO THE ADDRESS CONTAINED IN LOCATION "$LPERR". AFTER REPORTING THE ERROR. "$LPERR" IS SET BY EACH "SCOPE" CALL AND IS SET DIRECTLY DURING SOME SUBTESTS TO PROVIDE THE SMALLEST LOOP FOR LOOPING ON ERROR. IF SW9 = 0, THE PROGRAM WILL RETURN TO THE INSTRUCTION FOLLOWING THE ERROR CALL. (SEE SECTION 5.3 FOR MORE ON "LOOP ON ERROR").

3.2 INTERPRETING ERROR REPORTS

EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE ERROR CALL (ERRORPC). THESE TWO VALUES PINPOINT THE PLACE IN THE CODE THAT THE ERROR OCCURRED. BY REFERRING

TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE
COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST.
A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING
WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC
AND FUNCTIONS BEING TESTED.

EVERY ERROR REPORT ALSO TYPES AN ERROR MESSAGE
GIVING A VERBAL DESCRIPTION OF THE ERROR THAT HAS
BEEN DETECTED.

BY USING THE COMMENTS AND TEST DESCRIPTION FOUND IN
THE PROGRAM LISTING TO DETERMINE WHAT FUNCTION OR
LOGIC WAS BEING TESTED, THE OPERATOR CAN THEN REFER
TO THE ENGINEERING DRAWINGS TO ISOLATE THE PROBABLE
CAUSE FOR THE FAILURE.

3.3 SAMPLE ERROR REPORT

BELOW IS AN EXAMPLE OF AN ERROR WHICH COULD HAVE
OCCURRED DURING EXECUTION OF THE PROGRAM:

```
MEM. MGMT. REG. BITS NOT SET CORRECTLY
REGISTR WROTE   READ    READ-(BINARY)
ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC
177572  040000  060000  0110000000000000  000012  022060
```

WE SEE THAT THE ERROR OCCURRED IN TEST 12 AT LOACTION
022060. THE "REGISTR ADDRESS" TELLS US THAT WE WERE
TESTING MEMORY MANAGEMENT'S STATUS RGISTER 0 (SR0).
IN THE LISTING, THE TEST DESCRIPTION SAYS THAT THE
ERROR BITS (BITS <15:13>) OF SR0 WERE BEING SET AND
CLEARED INDIVIDUALLY. THE ERROR REPORT SAYS WE TRIED
TO SET BIT 14 BY WRITING "040000" TO SR0 BUT WHEN WE
READ IT BACK WE READ "060000". IT APPEARS THAT BIT 13 IS
STUCK AT "1" OR IT IS GETTING SET WHEN BIT 14 IS SET
TO "1". ERROR REPORTS BEFORE AND AFTER THIS ONE COULD
TELL US WHICH IS THE CASE.

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

THE PROGRAM IS FULLY ACT AND APT COMPATABLE
AND IS SUPPORTED UNDER THE XXDP PACKAGE.

4.2 END-OF-PASS MESSAGE

AT THE END OF EACH PASS OF THE PROGRAM THE PASS NUMBER
AND TOTAL NUMBER OF ERRORS SINCE THE LAST END-OF-PASS ARE
REPORTED IN THE END-OF-PASS MESSAGE. FOR EXAMPLE:

```
        END OF PASS #2  TOTAL ERRORS SINCE LAST REPORT 0
```

THAT WOULD INDICATE THAT PASS TWO WAS JUST COMPLETED
AND NO ERRORS WERE DETECTED DURING THAT PASS. BOTH
THE PASS NUMBER AND NUMBER OF ERRORS ARE DECIMAL NUMBERS.

4.3 T-BIT TRAPPING

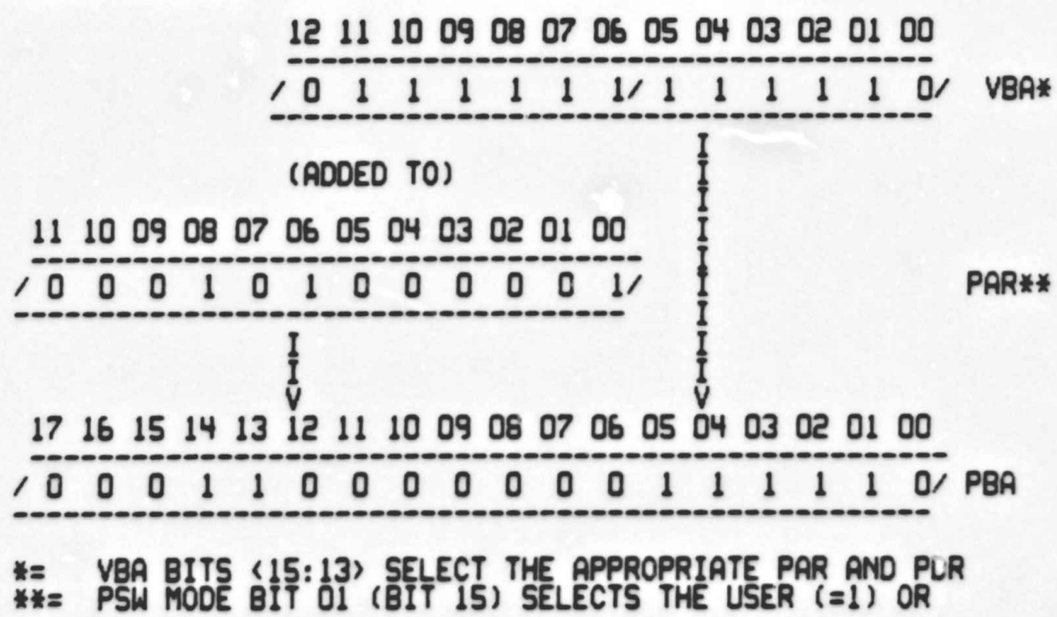
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET
BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS
BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT
TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH
REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE
MESSAGE "POWER FAILURE-RESTARTING" IS TYPED OUT AND
THE PROGRAM WILL RESTART EXECUTION AT "START:" (THE
VERY BEGINNING OF THE PROGRAM. IF THE SOFTWARE
SWITCH REGISTER IS BEING USED, ITS CONTENTS WILL BE
RESTORED. IF THERE IS A HARDWARE SWITCH REGISTER,
THERE IS NO WAY TO RESTORE THE VALUE OF THE SWITCH
REGISTER SO THE OPERATOR MUST RELOAD IT FROM THE CONSOLE.

4.5 PHYSICAL BUS ADDRESS CONSTRUCTION

BELOW IS A SIMPLIFIED DIAGRAM OF HOW THE MEMORY
MAMAGEMENT LOGIC CONSTRUCTS A PHYSICAL BUS ADDRESS
USING THE VIRTUAL ADDRESS AND THE PAGE ADDRESS REGISTER.
THE PAGE DESCRIPTOR REGISTER SELECTED WILL CONTAIN THE
PAGE EXPANSION, LENGTH, AND ACCESS INFORMATION.



*= VBA BITS (15:13) SELECT THE APPROPRIATE PAR AND PDR
**= PSW MODE BIT 01 (BIT 15) SELECTS THE USER (=1) OR

KERNEL (=0) SET OF PAR'S/PDR'S

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

FOLLOWING IS A LIST OF THE SUBROUTINES AND HANDLERS USED BY THIS PROGRAM THAT ARE NOT PROVIDED BY THE "SYSMAC PACKAGE". DETAILS OF THE SUBROUTINES UNIQUE TO THIS PROGRAM MAY BE FOUND IN THE PROGRAM LISTING. REFER TO THE "SYSMAC" DOCUMENT AND PROGRAM LISTING FOR THE OTHER ROUTINES.

1. TURN OFF T-BIT AND SAVE CURRENT PSW
2. TURN ON T-BIT AND RESTORE PREVIOUS PSW
3. SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4. READ AND COMPARE KERNEL AND USER PAR/PDR'S
5. CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

5.2 PROGRAM LISTING

A TABLE OF CONTENTS APPEARS AT THE BEGINNING OF THE LISTING WHICH CONTAINS THE NAMES OF EACH SECTION, SUBTEST, AND ROUTINE AND THE LINE NUMBERS CORRESPONDING TO THE START OF EACH.

FOLLOWING THIS SECTION OF DOCUMENTATION IS THE ACTUAL PROGRAM LISTING COMPLETE WITH SUBTEST DESCRIPTIONS AND "CODING COMMENTS".

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

WHEN AN ERROR OCCURS, ONE OF THE THINGS THAT'S IMPORTANT TO NOTE IS WHAT PASS THE ERROR OCCURRED ON. IF THE PASS NUMBER IS ODD AND IS THREE OR GREATER, THE ERROR MIGHT BE T-BIT SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 12 OF THE SWITCH REG. EQUAL TO "1" TO INHIBIT T-BIT TRAPPING. IF THE PASS NUMBER IS GREATER THAN ONE, THE ERROR MAY BE ITERATION SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 11 OF THE SWITCH REG. EQUAL TO "1" TO INHIBIT ITERATIONS. THESE HINTS SHOULD HELP YOU DETERMINE WHAT MAKES THE MACHINE FAIL AND WHEN.

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH REG. EQUAL TO "0", THEN YOU ARE ABLE TO LOOK AT ALL THE ERRORS THAT MAY BE RELATED TO THE FAULT YOU ARE DIAGNOSING. A FAULT IN AN EARLIER TEST MAY RESULT IN ERRORS DURING LATER TESTS WHICH MAY GIVE YOU MORE CLUES ABOUT THE NATURE OF THE FAULT. NOW USE THE METHOD OUTLINED IN SECTION 3.2 FOR EACH ERROR TO GATHER AS MUCH INFORMATION AS POSSIBLE.

```
                                NOW TO TEST YOUR IDEAS ON THE CAUSE OF THE FAILURE,
                                YOU MAY WANT TO SCOPE THIS ERROR CONDITION.  SET BIT 09
                                OF THE SWITCH REG. EQUAL TO "1" TO LOOP ON THE ERROR.
                                FOR AN EVEN TIGHTER SCOPE LOOP THE ERROR CALL CAN BE
                                REPLACED WITH A BRANCH (REFER TO COMMENTS BY ERROR CALLS
                                IN THE PROGRAM LISTING).

                                OR YOU COULD LOOP ON THE TEST BY EITHER SETTING BIT 14
                                OF THE SWITCH REG. EQUAL TO "1" OF BY SETTING BIT 08 OF THE
                                SWITCH REG. EQUAL TO "1" AND THEN SETTING THE TEST NUMBER
                                IN BITS 07-00 OF THE SWITCH REG.  YOU WILL PROBABLY WANT TO
                                INHIBIT ERROR TYPEOUTS BY SETTING BIT 13 OF THE SWITCH REG.
                                EQUAL TO "1".
```

```
                        .TITLE  MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
                        ;*COPYRIGHT (C) JAN-77
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            INHIBIT TRACE TRAP
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<7:0>
                        .SBTTL  BASIC DEFINITIONS

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

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

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

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
```

```
                    000100          PR2=    100             ;;PRIORITY LEVEL 2
                    000140          PR3=    140             ;;PRIORITY LEVEL 3
                    000200          PR4=    200             ;;PRIORITY LEVEL 4
                    000240          PR5=    240             ;;PRIORITY LEVEL 5
                    000300          PR6=    300             ;;PRIORITY LEVEL 6
                    000340          PR7=    340             ;;PRIORITY LEVEL 7

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

                                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                    100000          BIT15=  100000
                    040000          BIT14=  40000
                    020000          BIT13=  20000
                    010000          BIT12=  10000
                    004000          BIT11=  4000
                    002000          BIT10=  2000
                    001000          BIT09=  1000
                    000400          BIT08=  400
                    000200          BIT07=  200
                    000100          BIT06=  100
                    000040          BIT05=  40
                    000020          BIT04=  20
                    000010          BIT03=  10
                    000004          BIT02=  4
                    000002          BIT01=  2
                    000001          BIT00=  1
                                    .EQUIV  BIT09,BIT9
                                    .EQUIV  BIT08,BIT8
                                    .EQUIV  BIT07,BIT7
                                    .EQUIV  BIT06,BIT6
```

```
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

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

                        ;*KT11 VECTOR ADDRESS

000250                  MMVEC=  250

                        ;*KT11 STATUS REGISTER ADDRESSES

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

                        ;*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 "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

                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
```

```
                                ;*KERNEL "I" PAGE ADDRESS REGISTERS

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

                                        .EQUIV  SP,KSP
                                        .EQUIV  SP,USP
                                        .EQUIV  BIT4,TBIT
                                        .EQUIV  BIT6,WBIT
                                KERSTK= STACK
                                USESTK= STACK-200

                                ;*ADDITIONAL DEFINITIONS
                                ;*


                                .SBTTL  TRAP CATCHER

                                        .=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
                                        .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  020000                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
                                .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
                                ;HOOKS REQUIRED BY ACT11
                                        $SVPC=.                 ;SAVE PC
                                        .=46
                                        $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                                        .=52
                                        .WORD   0               ;;2)SET LOC.52 TO ZERO
                                        .=$SVPC                 ;; RESTORE PC
                                .SBTTL  APT PARAMETER BLOCK

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

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

                                $APTHD:
                                $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
                                $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
                                $TSTM:  .WORD   10      ;;RUN TIM OF LONGEST TEST
                                $PASTM: .WORD   20      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
                                $UNITM: .WORD   5       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
                                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102  000                     $TSTNM: .BYTE   0       ;;CONTAINS THE TEST NUMBER
001103  000                     $ERFLG: .BYTE   0       ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0       ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0       ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0       ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0       ;;CONTAINS TOTAL ERRORS DETECTED
001114  000                     $ITEMB: .BYTE   0       ;;CONTAINS ITEM CONTROL BYTE
001115  001                     $ERMAX: .BYTE   1       ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0       ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0       ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0       ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0       ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134  000                     $AUTOB: .BYTE   0       ;;AUTOMATIC MODE INDICATOR
001135  000                     $INTAG: .BYTE   0       ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR    ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560          ;;TTY KBD STATUS
001146  177562                  $TKB:   177562          ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000                  $REGAD: .WORD   0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001162  000000                  $REGO:  .WORD   0       ;;CONTAINS (($REGAD)+0)
001164  000000                  $REG1:  .WORD   0       ;;CONTAINS (($REGAD)+2)
001166  000000                  $REG2:  .WORD   0       ;;CONTAINS (($REGAD)+4)
001170  000000                  $REG3:  .WORD   0       ;;CONTAINS (($REGAD)+6)
001172  000000                  $REG4:  .WORD   0       ;;CONTAINS (($REGAD)+10)
001174  000000                  $REG5:  .WORD   0       ;;CONTAINS (($REGAD)+12)
001176  000000                  $TMP0:  .WORD   0       ;;USER DEFINED
001200  000000                  $TMP1:  .WORD   0       ;;USER DEFINED
001202  000000                  $TMP2:  .WORD   0       ;;USER DEFINED
001204  000000                  $TMP3:  .WORD   0       ;;USER DEFINED
001206  000000                  $TMP4:  .WORD   0       ;;USER DEFINED
001210  000000                  $TMP5:  .WORD   0       ;;USER DEFINED
001212  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001214  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001216  177607  000377              $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001222  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001223  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
```

```
  814  001224  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
  815                                  ;;*****************************************************************
  816                                  .SBTTL  APT MAILBOX-ETABLE
  817
  818                                  ;;*****************************************************************
  819                                  .EVEN
  820  001226                          $MAIL:                          ;;APT MAILBOX
  821  001226  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
  822  001230  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
  823  001232  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
  824  001234  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
  825  001236  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
  826  001240  000000                  $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
  827  001242  000000                  $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
  828  001244  000000                  $MSGLG: .WORD   AMSGLG   ;;MESSAGE LENGTH
  829  001246                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
  830  001246      000                 $ENV:   .BYTE   AENV     ;;ENVIRONMENT BYTE
  831  001247      000                 $ENVM:  .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
  832  001250  000000                  $SWREG: .WORD   ASWREG   ;;APT SWITCH REGISTER
  833  001252  000000                  $USWR:  .WORD   AUSWR    ;;USER SWITCHES
  834  001254  000000                  $CPUOP: .WORD   ACPUOP   ;;CPU TYPE,OPTIONS
  835                                  ;*                              BITS 15-11=CPU TYPE
  836                                  ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
  837                                  ;*                                      11/70=06,PDQ=07,Q=10
  838                                  ;*                              BIT 10=REAL TIME CLOCK
  839                                  ;*                              BIT  9=FLOATING POINT PROCESSOR
  840                                  ;*                              BIT  8=MEMORY MANAGEMENT
  841  001256      000                 $MAMS1: .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
  842  001257      000                 $MTYP1: .BYTE   AMTYP1   ;;MEM. TYPE,BLK#1
  843                                  ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
  844                                  ;*                                      900 NSEC CORE=001
  845                                  ;*                                      300 NSEC BIPOLAR=002
  846                                  ;*                                      500 NSEC MOS=003
  847  001260  000000                  $MADR1: .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
  848                                  ;*                               MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
  849  001262      000                 $MAMS2: .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
  850  001263      000                 $MTYP2: .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
  851  001264  000000                  $MADR2: .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
  852  001266      000                 $MAMS3: .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
  853  001267      000                 $MTYP3: .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
  854  001270  000000                  $MADR3: .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
  855  001272      000                 $MAMS4: .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
  856  001273      000                 $MTYP4: .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
  857  001274  000000                  $MADR4: .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
  858  001276  000000                  $VECT1: .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
  859  001300  000000                  $VECT2: .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
  860  001302  000000                  $BASE:  .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
  861  001304  000000                  $DEVM:  .WORD   ADEVM    ;;DEVICE MAP
  862  001306  000000                  $CDW1:  .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
  863  001310  000000                  $CDW2:  .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
  864  001312  000000                  $DDW0:  .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
  865  001314  000000                  $DDW1:  .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
  866  001316  000000                  $DDW2:  .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
  867  001320  000000                  $DDW3:  .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
  868  001322  000000                  $DDW4:  .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
  869  001324  000000                  $DDW5:  .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
```

```
  870  001326  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
  871  001330  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
  872  001332  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
  873  001334  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
  874  001336  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
  875  001340  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
  876  001342  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
  877  001344  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
  878  001346  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
  879  001350  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
  880
  881
  882  001352                        $ETEND:
  883
  884
  885  001352  000000                TESTNO: .WORD   0               ;HOLDS TEST NUMBER FOR TYPEOUTS
  886  001354  000000                WASR6:  .WORD   0               ;USED TO STORE THE STACK POINTER AFTER A TRAP
  887  001356  000000                TRAPPC: .WORD   0               ;USED TO STORE THE PC OF A TRAP OR ABORT
  888  001360  000000                TRAPPS: .WORD   0               ;USED TO STORE THE PS OF A TRAP OR ABORT
  889  001362  000000                WASSR0: .WORD   0               ;USED TO STORE CONTENTS OF SR0
  890  001364  000000                WASSR2: .WORD   0               ;USED TO STORE CONTENTS OR SR2
  891  001366  000000                TBIT?PS: .WORD  0               ;SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
  892  001370  000000                ANDADR: .WORD   0               ;HOLDS RESULT OF ADDRESSES BEING AND-ED
  893  001372  000000                ORADR:  .WORD   0               ;HOLDS RESULT OF ADDRESSES BEING OR-ED
  894  001374  000000                TONUM:  .WORD   0               ;HOLDS NUMBER OF TIME-OUTS
  895  001376  000000                VIRT1:  .WORD   0               ;HOLDS VIRTUAL ADDRESS TO BE CONVERTED
  896  001400  000000                VIRT2:  .WORD   0               ;   "       "       "
  897  001402  000000                PBALO:  .WORD   0               ;HOLDS BITS <15:00> OF PHYSICAL ADDRESS
  898  001404  000000                PBAHI:  .WORD   0               ;HOLDS BITS <17:16> OF PHYSICAL ADDRESS
```

```
                                .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


001406                          $ERRTB:

                                ;*ITEM 1
001406  041246                          EM1                     ;UNEXPECTED CPU TRAP TO LOC. 004
001410  044655                          DH1                     ;OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC
001412  050122                          DT1                     ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERPPC, 0
001414  050756                          DF1                     ;0,0,0,0,0

                                ;*ITEM 2
001416  041306                          EM2                     ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
001420  044725                          DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC
001422  050136                          DT2                     ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
001424  050763                          DF2                     ;0,0,0,0,0,0,0

                                ;*ITEM 3
001426  041355                          EM3                     ;PRIORITY BITS SET WRONG IN PSW
001430  045015                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001432  050156                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001434  050772                          DF3                     ;0,0,0,0

                                ;*ITEM 4
001436  041414                          EM4                     ;MODE BITS SET WRONG IN PSW
001440  045015                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001442  050156                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001444  050772                          DF3                     ;0,0,0,0

                                ;*ITEM 5
001446  041447                          EM5                     ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
001450  045015                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001452  050156                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001454  050772                          DF3                     ;0,0,0,0

                                ;*ITEM 6
001456  041522                          EM6                     ;KERNEL R6 CHANGED BY WRITING USER R6
001460  045015                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001462  050156                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001464  050772                          DF3                     ;0,0,0,0

                                ;*ITEM 7
001466  041567                          EM7                     ;A MEMORY MGMT. REG. TIMED OUT
001470  045055                          DH7                     ;ADDRESS TESTNO  ERRORPC
001472  050170                          DT7                     ;$REG0,TESTNO,$ERRPC,0
```

```
  955  001474  050776                    DF7                     ;0,0,0
  956
  957                                  ;*ITEM 10
  958  001476  041625                    EM10                    ;SUMMARY OF MEM. MGMT. REG. TIMEOUTS
  959  001500  045105                    DH10                    ;REGISTER-ADDRS  NUM. OF
  960                                                             ;AND-ED  OR-ED   TIMOUTS TESTNO ERRORPC
  961  001502  050200                    DT10                    ;ANDADR,ORADR,TONUM,TESTNO,$ERRPC,0
  962  001504  051001                    DF10                    ;0,0,1,0,0
  963
  964                                  ;*ITEM 11
  965  001506  041671                    EM11                    ;MEM. MGMT. REG. WOULD NOT CLEAR
  966  001510  045205                    DH11                    ;REGISTR READ    READ-(BINARY)
  967                                                             ;ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
  968  001512  050214                    DT11                    ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
  969  001514  051006                    DF11                    ;0,0,2,0,0
  970
  971                                  ;*ITEM 12
  972  001516  041731                    EM12                    ;MEM. MGMT. REG. BITS NOT SET CORRECTLY
  973  001520  045325                    DH12                    ;REGISTR WROTE   READ    READ
  974                                                             ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
  975  001522  050230                    DT12                    ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
  976  001524  051013                    DF12                    ;0,0,0,2,0,0
  977
  978                                  ;*ITEM 13
  979  001526  042000                    EM13                    ;SR0 EFFECTED BY WRITE TO PSW
  980  001530  045465                    DH13                    ;READ    TESTNO  ERRORPC
  981  001532  050246                    DT13                    ;$REG0,TESTNO,$ERRPC,0
  982  001534  051021                    DF13                    ;0,0,0
  983
  984                                  ;*ITEM 14
  985  001536  042035                    EM14                    ;SR1 DID NOT READ ALL ZEROS
  986  001540  045465                    DH13                    ;READ    TESTNO  ERRORPC
  987  001542  050246                    DT13                    ;$REG0,TESTNO,$ERRPC,0
  988  001544  051021                    DF13                    ;0,0,0
  989
  990                                  ;*ITEM 15
  991  001546  042070                    EM15                    ;DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
  992  001550  045325                    DH12                    ;REGISTER WROTE   READ    READ
  993                                                             ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO  ERRORPC
  994  001552  050230                    DT12                    ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
  995  001554  051013                    DF12                    ;0,0,0,2,0,0
  996
  997                                  ;*ITEM 16
  998  001556  042144                    EM16                    ;DUAL ADDRESSING BETWEEN PAR-PDR'S
  999  001560  045515                    DH16                    ;PAR-PDR PAR-PDR
 1000                                                             ;CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
 1001  001562  050256                    DT16                    ;$REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
 1002  001564  051024                    DF16                    ;0,0,0,0,0,0
 1003
 1004                                  ;*ITEM 17
 1005  001566  042206                    EM17                    ;PHYS. ADDR. FORMED READ WRONG IN MAINT. MODE
 1006  001570  045615                    DH17                    ;PHYSICAL VIRTUAL
 1007                                                             ;ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC
 1008  001572  050274                    DT17                    ;PBAL0,VIRT1,$REG4,TESTNO,$ERRPC,0
 1009  001574  051032                    DF17                    ;3,0,0,0,0
 1010
```

```
1011
1012  001576  042256                ;*ITEM 20
1013  001600  045705                        EM20            ;PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
                                      DH20            ;PHYSICL PAR 4   PAR 5
1014                                                  ;ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO
1015  001602  050310                DT20            ;PBAL0,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
1016  001604  051037                DF20            ;3,0,0,0,0,0,0,0
1017
1018                                ;*ITEM 21
1019  001606  042330                        EM21            ;W-BIT DID NOT GET SET IN PDR
1020  001610  046033                DH21            ;PDR     VIRTUAL
1021                                                  ;TESTED  ADDRESS TESTNO  ERRORPC
1022  001612  050332                DT21            ;$REG5,$REG3,TESTNO,$ERRPC,0
1023  001614  051047                DF21            ;0,0,0,0
1024
1025                                ;*ITEM 22
1026  001616  042365                        EM22            ;W-BIT SET IN MORE THAN ONE PDR
1027  001620  046113                DH22            ;PDR IN  PDR     VIRTUAL
1028                                                  ;ERROR   TESTED  ADDRESS TESTNO  ERRORPC
1029  001622  050344                DT22            ;$REG0,$REG5,$REG3,TESTNO,$ERRPC,0
1030  001624  051053                DF22            ;0,0,0,0,0
1031
1032                                ;*ITEM 23
1033  001626  042424                        EM23            ;W-BIT NOT CLEARED BY WRITING TO PDR
1034  001630  046212                DH23            ;PDR     TESTNO  ERRORPC
1035  001632  050360                DT23            ;$REG5,TESTNO,$ERRPC,0
1036  001634  051060                DF23            ;0,0,0
1037
1038                                ;*ITEM 24
1039  001636  042470                        EM24            ;WRITING SR0 SET W-BIT IN KIPDR7
1040  001640  046242                DH24            ;PDR WAS EXPECTD TESTNO  ERRORPC
1041  001642  050370                DT24            ;$REG2,$REG1,TESTNO,$ERRPC,0
1042  001644  051063                DF24            ;0,0,0,0
1043
1044                                ;*ITEM 25
1045  001646  042530                        EM25            ;W-BIT GOT SET DURING ODD ADDR. ABORT
1046  001650  046242                DH24            ;PDR WAS EXPECTD TESTNO  ERRORPC
1047  001652  050370                DT24            ;$REG2,$REG1,TESTNO,$ERRPC,0
1048  001654  051063                DF24            ;0,0,0,0
1049
1050                                ;*ITEM 26
1051  001656  042575                        EM26            ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1052  001660  046302                DH26            ;PDR 4   PSW     TESTNO  ERRORPC
1053  001662  050402                DT26            ;$REG2,$TMP0,TESTNO,$ERRPC,0
1054  001664  051063                DF24            ;0,0,0,0
1055
1056                                ;*ITEM 27
1057  001666  042645                        EM27            ;ACCESS ERROR DID NOT ABORT INSTRUCTION
1058  001670  046302                DH26            ;PDR 4   PSW     TESTNO  ERRORPC
1059  001672  050402                DT26            ;$REG2,$TMP0,TESTNO,$ERRPC,0
1060  001674  051063                DF24            ;0,0,0,0
1061
1062                                ;*ITEM 30
1063  001676  042714                        EM30            ;SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
1064  001700  046342                DH30            ;SR0 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1065  001702  050414                DT30            ;WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1066  001704  051067                DF30            ;0,0,0,0,0,0
```

```
1067
1068                                    ;*ITEM 31
1069  001706  042766                           EM31          ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1070  001710  046422                           DH31          ;SR2 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1071  001712  050432                           DT31          ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1072  001714  051067                           DF30          ;0,0,0,0,0,0
1073
1074                                    ;*ITEM 32
1075  001716  043033                           EM32          ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1076  001720  046502                           DH32          ;V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC
1077  001722  050450                           DT32          ;$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1078  001724  051067                           DF30          ;0,0,0,0,0,0
1079
1080                                    ;*ITEM 33
1081  001726  043114                           EM33          ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1082  001730  046562                           DH33          ;V.B.A.  KIPDR4  TESTNO  ERRORPC
1083  001732  050466                           DT33          ;$REG0,$REG4,TESTNO,$ERRPC,0
1084  001734  051063                           DF24          ;0,0,0,0
1085
1086                                    ;*ITEM 34
1087  001736  043177                           EM34          ;SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1088  001740  046622                           DH34          ;V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC
1089  001742  050500                           DT34          ;$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1090  001744  051067                           DF30          ;0,0,0,0,0,0
1091                                    ;*ITEM 35
1092  001746  042766                           EM31          ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1093  001750  046702                           DH35          ;V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC
1094  001752  050516                           DT35          ;$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1095  001754  051067                           DF30          ;0,0,0,0,0,0
1096
1097                                    ;*ITEM 36
1098  001756  042766                           EM31          ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1099  001760  046762                           DH36          ;SR2 WAS EXPECTD TESTNO  ERRORPC
1100  001762  050534                           DT36          ;WASSR2,$REG1,TESTNO,$ERRPC,0
1101  001764  051063                           DF24          ;0,0,0,0
1102
1103                                    ;*ITEM 37
1104  001766  043255                           EM37          ;SR0 OR SR2 CHANGED BY A SECOND ABORT
1105  001770  047022                           DH37          ;FIRST ABORT     SECOND ABORT
1106                                                          ;SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC
1107  001772  050546                           DT37          ;$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1108  001774  051067                           DF30          ;0,0,0,0,0,0
1109
1110                                    ;*ITEM 40
1111  001776  043322                           EM40          ;SR0 OR SR2 WAS NOT "RESET" BY A RESET
1112  002000  047137                           DH40          ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1113  002002  050564                           DT40          ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1114  002004  051063                           DF24          ;0,0,0,0
1115
1116                                    ;*ITEM 41
1117  002006  043371                           EM41          ;SR2 NOT TRACKING CORRECTLY
1118  002010  046762                           DH36          ;SR2 WAS EXPECTD TESTNO  ERROPC
1119  002012  050534                           DT36          ;WASSR2,$REG1,TESTNO,$ERRPC,0
1120  002014  051063                           DF24          ;0,0,0,0
1121
1122                                    ;*ITEM 42
```

ERROR POINTER TABLE

```
1123  002016  043424                EM42            ;DID NOT TRAP THRU KERNEL SPACE
1124  002020  047177                DH42            ;PSW WAS R6 WAS  TESTNO  ERRORPC
1125  002022  050576                DT42            ;$REG1,$REG2,TESTNO,$ERRPC,0
1126  002024  051063                DF24            ;0,0,0,0
1127
1128                              ;*ITEM 43
1129  002026  043463                EM43            ;KT ERROR SERVICED ON ODD ADDR. ERROR
1130  002030  046212                DH23            ;PDR     TESTNO  ERRORPC
1131  002032  050360                DT23            ;$REG5,TESTNO,$ERRPC,0
1132  002034  051060                DF23            ;0,0,0
1133
1134                              ;*ITEM 44
1135  002036  043530                EM44            ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
1136  002040  047237                DH44            ;EXPECTED        RECEIVED
1137                                                ;SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC
1138  002042  050610                DT44            ;$REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1139  002044  051067                DF30            ;0,0,0,0,0,0
1140
1141                              ;*ITEM 45
1142  002046  043576                EM45            ;ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)
1143  002050  047351                DH45            ;EXPECTED:
1144                                                ;PSW     PC        SR0     SR2
1145                                                ;170017  (3$+4)  020147  (3$)
1146                                                ;RECEIVED
1147                                                ;PSW     PC        SR0     SR2     TESTNO  ERRORPC
1148  002052  050626                DT45            ;$REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1149  002054  051067                DF30            ;0,0,0,0,0,0
1150
1151                              ;*ITEM 46
1152  002056  043653                EM46            ;MFPI INSTRUCTION PUSHED WRONG DATA
1153  002060  047546                DH46            ;DATA    DATA
1154                                                ;EXPECTD RECEIVD TESTNO  ERRORPC
1155  002062  050644                DT46            ;$REG0,$REG1,TESTNO,$ERRPC,0
1156  002064  051075                DF46            ;0,0,0,0
1157
1158                              ;*ITEM 47
1159  002066  043716                EM47            ;MTPI INSTRUCTION LOADED WRONG DATA
1160  002070  047546                DH46            ;DATA    DATA
1161                                                ;EXPECTD RECEIVD TESTNO  ERRORPC
1162  002072  050644                DT46            ;$REG0,$REG1,TESTNO,$ERRPC,0
1163  002074  051075                DF46            ;0,0,0,0
1164
1165                              ;*ITEM 50
1166  002076  043761                EM50            ;STACK NOT PUSHED BY MFPI-MTPI
1167  002100  047623                DH50            ;TESTNO  ERRORPC
1168  002102  050656                DT50            ;TESTNO,$ERRPC,0
1169  002104  051101                DF50            ;0,0
1170
1171                              ;*ITEM 51
1172  002106  044017                EM51            ;KERNEL PAGE ACCESSED IMSTEAD OF USER: MFPI-MTPI
1173  002110  047643                DH51            ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1174  002112  050664                DT51            ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1175  002114  051103                DF51            ;0,0,0,0
1176
1177                              ;*ITEM 52
1178  002116  044075                EM52            ;WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE
```

```
1179  002120  047703                    DH52                    ;PHYSICL PAR 4
1180                                                              ;ADDRESS V.B.A.  PAR 4   SR0 WAS SR2 WAS PSW    TESTNO
1181  002122  050676                    DT52                    ;PBAL0,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
1182  002124  051107                    DF52                    ;3,0,0,0,0,0,0,0
1183                              ;*ITEM 53
1184  002126  044153                    EM53                    ;MFPD INSTRUCTION PUSHED WRONG DATA
1185  002130  047546                    DH46                    ;DATA    DATA
1186                                                            ;EXPECTD RECEIVD TESTNO  ERRORPC
1187  002132  050644                    DT46                    ;$REG0,$REG1,TESTNO,$ERRPC,0
1188  002134  051075                    DF46                    ;0,0,0,0
1189
1190                              ;*ITEM 54
1191  002136  044216                    EM54                    ;STACK NOT PUSHED BY MFPD-MTPD
1192  002140  047623                    DH50                    ;TESTNO  ERRORPC
1193  002142  050656                    DT50                    ;TESTNO,$ERRPC,0
1194  002144  051101                    DF50                    ;0,0
1195
1196                              ;*ITEM 55
1197  002146  044254                    EM55                    ;PAR OR PDR WAS CHANGED BY A RESET
1198  002150  045205                    DH11                    ;REGISTR READ    READ-(BINARY)
1199                                                            ;ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
1200  002152  050214                    DT11                    ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
1201  002154  051006                    DF11                    ;0,0,2,0,0
1202
1203                              ;*ITEM 56
1204  002156  044312                    EM56                    ;ILLEGAL MODE 01 NOT ABORTED
1205  002160  047623                    DH50                    ;TESTNO  ERRORPC
1206  002162  050656                    DT50                    ;TESTNO,$ERRPC,0
1207  002164  051101                    DF50                    ;0,0
1208
1209                              ;*ITEM 57
1210  002166  044346                    EM57                    ;SR0 DID NOT REPORT ILLEGAL MODE 01 CORRECTLY
1211  002170  050021                    DH57                    ;SR0 WAS EXPECTD TESTNO  ERRORPC
1212  002172  050720                    DT57                    ;WASSR0,$REG1,TESTNO,$ERRPC,0
1213  002174  051117                    DF57                    ;0,0,0,0
1214
1215                              ;*ITEM 60
1216  002176  044423                    EM60                    ;PSW CHANGED BY AN RTI IN USER MODE
1217  002200  050061                    DH60                    ;PSW WAS EXPECTD TESTNO  ERRORPC
1218  002202  050732                    DT60                    ;$REG1,$REG2,TESTNO,$ERRPC,0
1219  002204  051123                    DF60                    ;0,0,0,0
1220
1221                              ;*ITEM 61
1222  002206  044466                    EM61                    ;MAINT MODE (SR0<8>) NOT DISABLED BY A RESET
1223  002210  047623                    DH50                    ;TESTNO  ERRORPC
1224  002212  050656                    DT50                    ;TESTNO,$ERRPC,0
1225  002214  051101                    DF50                    ;0,0
1226
1227                              ;*ITEM 62
1228  002216  044544                    EM62                    ;DATA INCORRECT AFTER A MAINT. MODE WRITE
1229  002220  045015                    DH3                     ;WROTE   READ    TESTNO  ERRORPC
1230  002222  050156                    DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
1231  002224  050772                    DF3                     ;0,0,0,0
1232
1233                              ;*ITEM 63
1234  002226  044615                    EM63                    ;SOURCE RELOCATED IN MAINT. MODE
```

```
1235  002230  044725                  DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2    TESTNO  ERRORPC
1236  002232  050136                  DT2                     ;TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
1237  002234  050763                  DF2                     ;0,0,0,0,0,0,0
1238
1239                                ;*ITEM 64
1240  002236  042766                  EM31                    ;SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR.
1241  002240  046762                  DH36                    ;SR2 WAS EXPECTD TESTNO  ERRORPC
1242  002242  050744                  DT64                    ;WASSR2,$REG4,TESTNO,$ERRPC,0
1243  002244  051063                  DF24                    ;0,0,0,0
1244
```

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

                                .SBTTL  CPU TRAP HANDLER ROUTINE
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
                                ;*      "ERRVEC" (LOC. 004).  IF THIS SUBROUTINE IS ENTERED BY A
                                ;*      SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
                                ;*      EXECUTED.
                                ;*
                                ;;*****************************************************************
002246  005227                  TIMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
002250  177777                  TIMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
002252  001401                          BEQ     1$              ;BRANCH IF FIRST TIME IN
002254  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE I FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
002256  012637  001356          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
002262  012637  001360                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
002266  010637  001354                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
002272  104001                          ERROR   1               ;UNEXPECTED TRAP OR ABORT TO LOC. 4
002274  012737  177777  002250        MOV     #-1,TIMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
002302  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
002306  013746  001356                  MOV     TRAPPC,-(KSP)
002312  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT


                                .SBTTL  MEMORY MANAGEMENT TRAP HANDLER ROUTINE
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
                                ;*      TRAPS AND ABORTS THRU "MMVEC" (LOC. 250).  IF THIS SUBROUTINE IS
                                ;*      ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
                                ;*      HALT IS EXECUTED.
                                ;*
                                ;;*****************************************************************
002314  005227                  MGMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
002316  177777                  MGMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
002320  001401                          BEQ     1$              ;BRANCH IF FIRST TIME IN
002322  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE I FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
002324  012637  001356          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
002330  012637  001360                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
002334  010637  001354                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
002340  013737  177572  001362        MOV     SR0,WASSR0      ;SAVE CONTENTS OF KT STATUS REG. 0
002346  013737  177576  001364        MOV     SR2,WASSR2      ;SAVE CONTENTS OF KT STATUS REG. 2
002354  042737  160000  177572        BIC     #160000,SR0     ;CLEAR ERROR BITS IN STATUS REG 0
002362  104002                          ERROR   2               ;UNEXPECTED TRAP OR ABORT TO LOC. 250
002364  012737  177777  002316        MOV     #-1,MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
002372  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
002376  013746  001356                  MOV     TRAPPC,-(KSP)
```

```
1301  002402  000006                RTT                     ;RETURN FROM INTERRUPT OR ABORT.
1302
```

```
1303                                            .SBTTL
1304                                            .SBTTL  *****   STARTING POINT OF TEST  *****
1305                                            .SBTTL  *****   STARTING ADDRESS OF 200 *****
1306            020000                          .=20000
1307
1308    020000                          START:
1309                                    .SBTTL  INITIALIZE THE COMMON TAGS
1310                                    ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1311    020000  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1312    020004  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1313    020006  022706  001140                  CMP     #SWR,R6 ;;DONE?
1314    020012  001374                          BNE     .-6             ;;LOOP BACK IF NO
1315    020014  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1316                                    ;;INITIALIZE A FEW VECTORS
1317    020020  012737  034470  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1318    020026  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1319    020034  012737  034750  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1320    020042  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1321    020050  012737  040732  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1322    020056  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
1323    020064  012737  041020  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1324    020072  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1325    020100  013737  034236  034230          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1326    020106  005037  001212                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1327    020112  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1328    020116  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1329                                    ;;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "$RTRN", IN
1330                                    ;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT".
1331    020124  012737  034454  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
1332    020132  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
1333    020140  012737  000002  034454          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
1334    020146  012737  020174  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
1335    020154  005046                          CLR     -(SP)           ;;DUMMY PS
1336    020156  012746  020164                  MOV     #64$,-(SP)      ;;AND PC
1337    020162  000006                          RTT                     ;;TRY THE RTT
1338    020164  012737  000006  034454  64$:    MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
1339    020172  000402                          BR      66$
1340    020174  062706  000010          65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
1341    020200  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
1342    020206  005037  034462                  CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
1343    020212  012737  020212  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1344    020220  012737  020220  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1345                                    ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1346                                    ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1347    020226  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1348    020232  012737  020266  000004          MOV     #67$,@#ERRVEC   ;;SET UP ERROR VECTOR
1349    020240  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1350    020246  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1351    020254  022777  177777  160656          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1352    020262  001012                          BNE     69$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1353                                                                    ;;AND  THE HARDWARE SWR IS NOT = -1
1354    020264  000403                          BR      68$             ;;BRANCH IF NO TIMEOUT
1355    020266  012716  020274          67$:    MOV     #68$,(SP)       ;;SET UP FOR TRAP RETURN
1356    020272  000002                          RTI
1357    020274  012737  000176  001140  68$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1358    020302  012737  000174  001142          MOV     #DISPREG,DISPLAY
```

```
1359  020310  012637  000004          69$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1360
1361  020314  005037  001234                  CLR     $PASS           ;;CLEAR PASS COUNT
1362  020320  132737  000200  001247          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
1363  020326  001403                          BEQ     70$             ;;YES,USE NON-APT SWITCH
1364  020330  012737  001250  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
1365  020336                          70$:
1366                                  .SBTTL  TYPE PROGRAM NAME
1367                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1368  020336  005227  177777                  INC     #-1             ;;FIRST TIME?
1369  020342  001054                          BNE     71$             ;;BRANCH IF NO
1370  020344  022737  034410  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
1371  020352  001450                          BEQ     71$             ;;BRANCH IF YES
1372  020354  104401  020422                  TYPE    ,72$            ;;TYPE ASCIZ STRING
1373                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1374  020360  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1375  020364  001012                          BNE     73$             ;;BRANCH IF YES
1376  020366  123727  001246  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
1377  020374  001406                          BEQ     73$             ;;BRANCH IF YES
1378  020376  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1379  020404  001005                          BNE     74$             ;;BRANCH IF NO
1380  020406  104407                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1381  020410  000403                          BR      74$
1382  020412  112737  000001  001134  73$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1383  020420                          74$:
1384  020420  000425                          BR      71$             ;;GET OVER THE ASCIZ
1385                                  ;;72$:  .ASCIZ  <CRLF>#MD-11-DFKTH-A  11/34 MEMORY MGMT. DIAG.#<CRLF>
1386  020474                          71$:
1387
1388  020474                          LOOP:
1389  020474  012706  001100                  MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
1390  020500  012737  002246  000004          MOV     #TIMERR,ERRVEC  ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1391  020506  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1392  020514  012737  002314  000250          MOV     #MGMERR,MMVEC   ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1393  020522  012737  000340  000252          MOV     #340,MMVEC+2    ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1394  020530  012700  177777                  MOV     #-1,R0          ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1395  020534  010037  002250                  MOV     R0,TIMFLG       ;INITIALIZE CPU ERROR FLAG
1396  020540  010037  002316                  MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1397  020544  012737  000340  001366          MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1398  020552  005037  177572                  CLR     SR0             ;BE SURE MEM. MGMT IS OFF TO START WITH
1399
```

```
1400
1401
1402                                    ;;*****************************************************************
1403                                    ;*TEST 1        PSW PRIORITY BIT TEST
1404                                    ;*
1405                                    ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <7:5> "PRIORITY BITS"
1406                                    ;*      TO SEE THAT SOME OF THE BASIC "DATA PATH" LOGIC IS WORKING.
1407                                    ;*
1408                                    ;;*****************************************************************
1409  020556  000004                    TST1:   SCOPE
1410  020560  012737  020570  001110    1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1411  020566  005000                            CLR     R0              ;INITIALIZE R0 WITH PRIORITY=0 DATA
1412  020570  005001                    2$:     CLR     R1              ;PREPARE R1 TO ACCEPT DATA READ
1413  020572  106400                            MTPS    R0              ;WRITE PRIORITY BITS IN THE PSW
1414  020574  106701                            MFPS    R1              ;READ BACK THE LOW BYTE OF PSW
1415  020576  042701  177437                    BIC     #177437,R1      ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
1416  020602  020001                            CMP     R0,R1           ;WAS CORRECT PRIORITY SET IN THE PSW?
1417  020604  001401                            BEQ     3$              ;BRANCH IF YES
1418  020606  104003                            ERROR   3               ;PRIORITY BITS SET WRONG IN PSW
1419                                                                    ;FOR TIGHTER SCOPE LOOP
1420                                                                    ;REPLACE ERROR CALL WITH
1421                                                                    ;"BR 2$" = 000770
1422  020610  062700  000040            3$:     ADD     #40,R0          ;CHANGE DATA TO NEXT PRIORITY
1423  020614  022700  000400                    CMP     #400,R0         ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
1424  020620  001363                            BNE     2$              ;BRANCH IF NO
1425  020622  012737  020560  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1426
1427                                    ;;*****************************************************************
1428                                    ;*TEST 2        PSW MODE BIT TEST
1429                                    ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> "MODE BITS"
1430                                    ;*      TO FURTHER CHECK THE BASIC CPU DATA PATHS
1431                                    ;*
1432                                    ;;*****************************************************************
1433  020630  000004                    TST2:   SCOPE
1434  020632  012737  020642  001110    1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1435  020640  005000                            CLR     R0              ;INITIALIZE R0 WITH MODE BITS = 0000
1436  020642  005037  177776            2$:     CLR     PSW             ;INITIALIZE PSW
1437  020646  050037  177776                    BIS     R0,PSW          ;BIT SET THE PSW MODE BITS WITH R0
1438  020652  013701  177776                    MOV     PSW,R1          ;READ BACK THE CONTENTS OF THE PSW
1439  020656  042701  007777                    BIC     #007777,R1      ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
1440  020662  020001                            CMP     R0,R1           ;WERE THE MODE BITS SET CORRECTLY?
1441  020664  001403                            BEQ     3$              ;BRANCH IF YES
1442  020666  005037  177776                    CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
1443  020672  104004                            ERROR   4               ;MODE BITS SET WRONG IN PSW
1444                                                                    ;FOR TIGHTER SCOPE LOOP
1445                                                                    ;REPLACE ERROR CALL WITH
1446                                                                    ;"BR 2$" = 000763
1447  020674  062700  010000            3$:     ADD     #10000,R0       ;CHANGE MODE BIT DATA
1448  020700  001360                            BNE     2$              ;BRANCH IF STILL MORE COMBINATIONS
1449  020702  012737  020632  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1450  020710  005037  177776                    CLR     PSW             ;RESET PSW BEFORE LEAVING
1451
1452                                    ;;*****************************************************************
1453                                    ;*TEST 3        BYTE ADDRESSING TEST FOR PSW
1454                                    ;*
1455                                    ;*      THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
```

```
                        ;*      AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
                        ;*      THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
                        ;*
                        ;;*****************************************************************
020714  000004         TST3:   SCOPE
020716  012737  020724  001110 1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
020724  005037  177776 2$:     CLR     PSW             ;CLEAR THE PSW
020730  012700  000360         MOV     #360,R0         ;PUT THE HIGH BYTE DATA INTO R0
020734  110037  177777         MOVB    R0,PSW+1        ;WRITE THE HIGH BYTE OF THE PSW
020740  013701  177776         MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
020744  042701  007437         BIC     #007437,R1      ;MASK OFF THE T & CC BITS
020750  000300                 SWAB    R0              ;GET DATA WRITTEN IN HIGH BYTE OF R0
020752  020001                 CMP     R0,R1           ;WAS THE PSW WRITTEN TO CORRECTLY
020754  001403                 BEQ     3$              ;BRANCH IF YES
020756  005037  177776         CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
020762  104005                 ERROR   5               ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
                                                       ;FOR TIGHTER SCOPE LOOP
                                                       ;REPLACE ERROR CALL WITH
                                                       ;"BR 2$" = 000760
020764  012737  020772  001110 3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
020772  005037  177776 4$:     CLR     PSW             ;CLEAR THE PSW
020776  012700  000340         MOV     #340,R0         ;PUT THE LOW BYTE DATA INTO R0
021002  110037  177776         MOVB    R0,PSW          ;WRITE THE LOW BYTE OF THE PSW
021006  013701  177776         MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
021012  042701  007437         BIC     #007437,R1      ;MASK OFF THE T&CC BITS
021016  020001                 CMP     R0,R1           ;WAS PSW WRITTEN TO CORRECTLY
021020  001403                 BEQ     5$              ;BRANCH IF YES
021022  005037  177776         CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
021026  104005                 ERROR   5               ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
                                                       ;FOR TIGHTER SCOPE LOOP
                                                       ;REPLACE ERROR CALL WITH
                                                       ;"BR 2$" = 000736
021030  012737  020716  001110 5$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                        ;;*****************************************************************
                        ;*TEST 4        TEST AND SETUP OF STACK POINTERS
                        ;*
                        ;*      THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
                        ;*      REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
                        ;*      EACH OTHER.  KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
                        ;*      KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
                        ;*
                        ;;*****************************************************************
021036  000004         TST4:   SCOPE
021040  005037  177776         CLR     PSW             ;GO TO KERNEL MODE
021044  012706  001100         MOV     #KERSTK,KSP     ;SET KERNEL STACK POINTER TO 1100
021050  012737  140000  177776 MOV     #140000,PSW     ;GO TO USER MODE
021056  012706  000700         MOV     #USESTK,USP     ;SET USER STACK POINTER TO 700
021062  005037  177776         CLR     PSW             ;BACK TO KERNEL MODE
021066  022706  001100         CMP     #KERSTK,KSP     ;IS KERNEL R6 STILL 1100?
021072  001404                 BEQ     TST5            ;;BRANCH IF KERNEL R6 IS OKAY
021074  012700  001100         MOV     #KERSTK,R0      ;SAVE DATA WRITTEN FOR ERROR REPORT
021100  010601                 MOV     KSP,R1          ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
021102  104006                 ERROR   6               ;KERNEL R6 CHANGED BY WRITING USER R6
                                                       ;FOR TIGHTER SCOPE LOOP
                                                       ;REPLACE ERROR CALL WITH
```

```
                                                        ;000756
                                ;;*********************************************************************
                                ;*
                                ;*      THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
                                ;*      MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,KERNEL & USER PAR/PDR'S).
                                ;*      EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
                                ;*      AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
                                ;*      OUT DURING THAT TEST IS GIVEN.  THE RESULTS OF "AND-ING" AND "OR-ING"
                                ;*      THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
                                ;*      STUCK AT 0 OR 1.  THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
                                ;*      THINGS BEING CHECKED.
                                ;*
                                ;;*********************************************************************


                                ;;*********************************************************************
                                ;*TEST 5        SR0,SR1,SR2 TIMEOUT TEST
                                ;*
                                ;*      THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
                                ;*      0,1, AND 2.  STATUS REG. 1 IS NOT USED BUT SHOULD STILL
                                ;*      RESPOND TO ITS UNIBUS ADDRESS.  DATA WILL BE WRITTEN OR READ
                                ;*      FROM THESE REGISTERS IN LATER TESTS, THIS TEST JUST CHECK
                                ;*      FOR A RESPONSE.
                                ;*
                                ;;*********************************************************************
021104  000004                  TST5:   SCOPE

021106  012737  021150  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
021114  012737  021206  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
021122  012700  177572                  MOV     #SR0,R0         ;LOAD R0 WITH ADDRESS OF FIRST REG.
021126  012701  000003                  MOV     #3,R1           ;LOAD R1 WITH THE LOOP COUNT
021132  012737  177777  001370          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDRS. LOC.
021140  005037  001372                  CLR     ORADR           ;INITIALIZE "OR" OF ADDRS. LOC.
021144  005037  001374                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
021150  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A STATUS REGISTER
                                                                ;IF IT TIMES OUT GO TO 5$
021152  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT ADDRESS IN R0
021156  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
021160  012737  021106  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
021166  005737  001374                  TST     TONUM           ;DID ANY OF THE STATUS REG.S TIMEOUT?
021172  001401                          BEQ     4$              ;BRANCH IF NO
021174  104010                          ERROR   10              ;SUMMARY OF STATUS REG. TIMEOUTS
021176  012737  002246  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021204  000414                          BR      TST6            ;;GO TO NEXT TEST

021206  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
021212  104007                          ERROR   7               ;ONE OF THE STATUS REGS. TIMED OUT
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000756
021214  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021216  050237  001372                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
021222  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
021224  040237  001370                  BIC     R2,ANDADR
021230  005237  001374                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
```

```
                                        BR      3$              ;BRANCH BACK TO TEST THE NEXT ADDR.

                                ;;*****************************************************************
                                ;*TEST 6         KERNEL PAR'S TIMEOUT TEST
                                ;*
                                ;*      THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
                                ;*      REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
                                ;*      RESPONDS TO THEIR ADDRESSES.
                                ;*
                                ;;*****************************************************************
021236  000004                TST6:   SCOPE

021240  012737  021302  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
021246  012737  021340  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
021254  012700  172340                  MOV     #KIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
021260  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
021264  012737  177777  001370          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
021272  005037  001372                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
021276  005037  001374                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
021302  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A KIPAR
                                                                ;IF IT TIMES OUT, WILL GO TO 5$
021304  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT KIPAR ADDRESS IN R0
021310  077104                  4$:     SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
021312  012737  021240  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
021320  005737  001374                  TST     TONUM           ;DID ANY OF THE KIPARS TIME OUT?
021324  001401                          BEQ     4$              ;BRANCH IF NO
021326  104010                          ERROR   10              ;SUMMARY OF KIPAR TIMEOUTS
021330  012737  002246  000004          MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021336  000414                          BR      TST7            ;;GO TO NEXT TEST

021340  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
021344  104007                          ERROR   7               ;ONE OF THE KIPARS TIMED OUT
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000756
021346  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021350  050237  001372                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
021354  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
021356  040237  001370                  BIC     R2,ANDADR
021362  005237  001374                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
021366  000746                          BR      3$              ;BRANCH BACK TO TEST THE NEXT KIPAR

                                ;;*****************************************************************
                                ;*TEST 7         KERNEL PDR'S TIMEOUT TEST
                                ;*
                                ;*      THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
                                ;*      REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
                                ;*      RESPONDS TO THEIR ADDRESSES.
                                ;*
                                ;;*****************************************************************
021370  000004                TST7:   SCOPE

021372  012737  021434  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
021400  012737  021472  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
021406  012700  172300                  MOV     #KIPDR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
021412  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
```

```
1624  021416  012737  177777  001370          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1625  021424  005037  001372                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
1626  021430  005037  001374                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1627  021434  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A KIPDR
1628                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1629  021436  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT KIPDR ADDRESS IN R0
1630  021442  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1631  021444  012737  021372  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1632  021452  005737  001374                  TST     TONUM           ;DID ANY OF THE KIPDRS TIME OUT?
1633  021456  001401                          BEQ     4$              ;BRANCH IF NO
1634  021460  104010                          ERROR   10              ;SUMMARY OF KIPDR TIMEOUTS
1635  021462  012737  002246  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1636  021470  000414                          BR      TST10           ;;GO TO NEXT TEST
1637
1638  021472  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
1639  021476  104007                          ERROR   7               ;ONE OF THE KIPDRS TIMED OUT
1640                                                                  ;FOR TIGHTER SCOPE LOOP
1641                                                                  ;REPLACE ERROR CALL WITH
1642                                                                  ;"BR 2$" = 000756
1643  021500  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1644  021502  050237  001372                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1645  021506  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1646  021510  040237  001370                  BIC     R2,ANDADR
1647  021514  005237  001374                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
1648  021520  000746                          BR      3$              ;BRANCH BACK TO TEST THE NEXT KIPDR
1649
1650                                  ;;**************************************************************
1651                                  ;*TEST 10       USER PAR'S TIMEOUT TEST
1652                                  ;*
1653                                  ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1654                                  ;*      REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1655                                  ;*      RESPONDS TO THEIR ADDRESSES.
1656                                  ;*
1657                                  ;;**************************************************************
1658  021522  000004          TST10:  SCOPE
1659
1660  021524  012737  021566  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1661  021532  012737  021624  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
1662  021540  012700  177640                  MOV     #UIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1663  021544  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
1664  021550  012737  177777  001370          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1665  021556  005037  001372                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
1666  021562  005037  001374                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1667  021566  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A UIPAR
1668                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1669  021570  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT UIPAR ADDRESS IN R0
1670  021574  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1671  021576  012737  021524  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1672  021604  005737  001374                  TST     TONUM           ;DID ANY OF THE UIPARS TIME OUT?
1673  021610  001401                          BEQ     4$              ;BRANCH IF NO
1674  021612  104010                          ERROR   10              ;SUMMARY OF UIPAR TIMEOUTS
1675  021614  012737  002246  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1676  021622  000414                          BR      TST11           ;;GO TO NEXT TEST
1677
1678  021624  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
1679  021630  104007                          ERROR   7               ;ONE OF THE UIPARS TIMED OUT
```

```
1680                                                            ;FOR TIGHTER SCOPE LOOP
1681                                                            ;REPLACE ERROR CALL WITH
1682                                                            ;"BR 2$" = 000756
1683  021632  010002                    MOV     R0,R2          ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1684  021634  050237  001372            BIS     R2,ORADR       ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1685  021640  005102                    COM     R2             ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1686  021642  040237  001370            BIC     R2,ANDADR
1687  021646  005237  001374            INC     TONUM          ;INCREMENT THE TIMEOUT COUNTER
1688  021652  000746                    BR      3$             ;BRANCH BACK TO TEST THE NEXT UIPAR
1689
1690                                ;;*****************************************************************
1691                                ;*TEST 11       USER PDR'S TIMEOUT TEST
1692                                ;*
1693                                ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
1694                                ;*      REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
1695                                ;*      RESPONDS TO THEIR ADDRESSES.
1696                                ;*
1697                                ;*****************************************************************
1698  021654  000004                TST11:  SCOPE
1699
1700  021656  012737  021720  001110  1$:   MOV     #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
1701  021664  012737  021756  000004        MOV     #5$,@#4        ;SET TIMEOUT VECTOR TO 5$
1702  021672  012700  177600                MOV     #UIPDR0,R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
1703  021676  012701  000010                MOV     #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
1704  021702  012737  177777  001370        MOV     #-1,ANDADR     ;INITIALIZE "AND" OF ADDR. LOC
1705  021710  005037  001372                CLR     ORADR          ;INITIALIZE "OR" OF ADDR. LOC.
1706  021714  005037  001374                CLR     TONUM          ;INITIALIZE "TIMEOUTS" COUNTER
1707  021720  005710                2$:     TST     (R0)           ;TRY ADDRESSING A UIPDR
1708                                                               ;IF IT TIMES OUT, WILL GO TO 5$
1709  021722  062700  000002        3$:     ADD     #2,R0          ;PUT NEXT UIPDR ADDRESS IN R0
1710  021726  077104                        SOB     R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1711  021730  012737  021656  001110        MOV     #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1712  021736  005737  001374                TST     TONUM          ;DID ANY OF THE UIPDRS TIME OUT?
1713  021742  001401                        BEQ     4$             ;BRANCH IF NO
1714  021744  104010                        ERROR   10             ;SUMMARY OF UIPDR TIMEOUTS
1715  021746  012737  002246  000004  4$:   MOV     #TIMERR,@#4    ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1716  021754  000414                        BR      TST12          ;;GO TO NEXT TEST
1717
1718  021756  062706  000004        5$:     ADD     #4,KSP         ;CLEAN UP THE STACK
1719  021762  104007                        ERROR   7              ;ONE OF THE UIPDRS TIMED OUT
1720                                                               ;FOR TIGHTER SCOPE LOOP
1721                                                               ;REPLACE ERROR CALL WITH
1722                                                               ;"BR 2$" = 000756
1723  021764  010002                    MOV     R0,R2          ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1724  021766  050237  001372            BIS     R2,ORADR       ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1725  021772  005102                    COM     R2             ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1726  021774  040237  001370            BIC     R2,ANDADR
1727  022000  005237  001374            INC     TONUM          ;INCREMENT THE TIMEOUT COUNTER
1728  022004  000746                    BR      3$             ;BRANCH BACK TO TEST THE NEXT UIPDR
1729
1730                                ;;*****************************************************************
1731                                ;*TEST 12       SR0(15:13) BIT TEST & SR2 TEST
1732                                ;*
1733                                ;*      THIS TEST CHECKS BITS (15:13) OF STATUS REGISTER 0 TO SEE
1734                                ;*      THAT EACH CAN BE SET AND CLEARED AND THAT A "RESET" WILL
1735                                ;*      CLEAR ALL OF THEM.  A TEST OF THESE THREE ERROR BITS CHECKS
```

```
                                        ;*      PART OF SR0, THE SR0 MUX AND THE KTMUX.  THE REST OF THE
                                        ;*      BITS IN SR0 WILL BE CHECKED LATER.
                                        ;*      ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
                                        ;*      OFF BUT LOCKS UP WHEN ANY OF SR0 ERROR BITS SET.
                                        ;*
                                        ;;*****************************************************************
022006  000004                          TST12:  SCOPE

022010  012700  177572                  1$:     MOV     #SR0,R0         ;LOAD ADDRESS OF SR0 INTO R0
022014  012710  160000                          MOV     #160000,(R0)    ;SET BITS <15:13> IN SR0 (ERROR BITS)
022020  000005                                 RESET                   ;ISSUE AND "INIT" SIGNAL
022022  011001                                 MOV     (R0),R1         ;READ SR0 INTO R1 TO SEE IF CLEAR
022024  001404                                 BEQ     2$              ;BRANCH IF SR0<15:13> CLEARED BY "INIT"
022026  104011                                 ERROR   11              ;SR0<15:13> NOT CLEARED BY A "RESET"
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 1$" = 000770
022030  012737  022036  001110                 MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
022036  013737  177576  001364  2$:     MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
022044  012701  022036                          MOV     #2$,R1          ;LOAD EXPECTED CONTENTS INTO R1
022050  020137  001364                          CMP     R1,WASSR2       ;IS SR2 TRACKING?
022054  001401                                 BEQ     3$              ;BRANCH IF YES
022056  104041                                 ERROR   41              ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 2$" = 000767
022060  012737  022076  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
022066  012701  100000                          MOV     #BIT15,R1       ;PUT DATA TO BE WRITTEN IN R1
022072  012703  000003                          MOV     #3,R3           ;SETUP R3 AS A LOOP COUNTER
022076  005010                          4$:     CLR     (R0)            ;CLEAR SR0
022100  050110                          5$:     BIS     R1,(R0)         ;SET ONE OF THE ERROR BITS IN SR0
022102  011002                                 MOV     (R0),R2         ;READ SR0 INTO R2
022104  020102                                 CMP     R1,R2           ;DID RIGHT ERROR BIT GET SET?
022106  001401                                 BEQ     6$              ;BRANCH IF YES
022110  104012                                 ERROR   12              ;BITS WERE SET WRONG IN SR0
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 4$" = 000772
022112  012704  022100                  6$:     MOV     #5$,R4          ;LOAD EXPECTED CONTENTS OF SR2 IN R4
022116  013737  177576  001364                 MOV     SR2,WASSR2      ;READ SR2
022124  020437  001364                          CMP     R4,WASSR2       ;DID SR2 LOCK UP WHEN ERROR
                                                                        ;BIT SET IN SR1?
022130  001401                                 BEQ     7$              ;BRANCH IF YES
022132  104064                                 ERROR   64              ;SR2 DID NOT LOCK UP
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 4$" = 000761
022134  006001                          7$:     ROR     R1              ;CHANGE DATA TO CHECK NEXT ERROR BIT
022136  077321                                 SOB     R3,4$           ;LOOP BACK UNTIL <15:13> ALL TESTED
022140  005010                                 CLR     (R0)            ;CLEAR SR0 BEFORE LEAVING
022142  012737  022010  001110                 MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                                        ;;*****************************************************************
                                        ;*TEST 13       SR0 & PSW DUAL ADDRESSING TEST
                                        ;*
                                        ;*      THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
```

```
                        ;*      VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
                        ;*      TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
                        ;*      IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE.  THIS IS TO
                        ;*      SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
                        ;*
                        ;;*****************************************************************
022150  000004         TST13:  SCOPE

022152  005037  177776  1$:     CLR     PSW             ;CLEAR THE PSW
022156  005037  177572         CLR     SR0             ;CLEAR STATUS REGISTER 0
022162  106427  000340         MTPS    #340            ;SET PRIORITY 7 IN LOW BYTE OF PSW
022166  013700  177572         MOV     SR0,R0          ;READ STATUS REGISTER 0
022172  001401                 BEQ     2$              ;BRANCH IF IT WAS STILL 0
022174  104013                 ERROR   13              ;SR0 EFFECTED BY A WRITE TO THE PSW
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 1$" = 000767
022176  005037  177572  2$:     CLR     SR0             ;BE SURE SR0 IS 0 BEFORE LEAVING
022202  005037  177776         CLR     PSW             ;BE SURE PSW IS 0 BEFORE LEAVING

                        ;;*****************************************************************
                        ;*TEST 14       TEST THAT SR1 READS ALL ZEROS
                        ;*
                        ;*      THIS TESTS CHECKS THAT EVEN THOUGH STATUS REGISTER 1
                        ;*      IS NON-EXISTENT, ITS ADDRESS SHOULD RESPOND WITH ALL ZEROS,
                        ;*      THEREBY CHECK ANOTHER PORTION OF THE ADDRESS DETECTION LOGIC.
                        ;*
                        ;;*****************************************************************
022206  000004         TST14:  SCOPE
022210  012700  177777         MOV     #-1,R0          ;FILL R0 WITH ALL ONES
022214  013700  177574         MOV     SR1,R0          ;READ SR1 INTO R0
022220  001401                 BEQ     TST15           ;;BRANCH IF SR1 READS ALL ZEROS
022222  104014                 ERROR   14              ;SR1 DID NOT READ ALL ZEROS
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;000772

                        ;;*****************************************************************
                        ;*TEST 15       BIT TEST OF KERNEL & USER PAR'S
                        ;*
                        ;*      THE FOLLOWING TEST CHECKS THE BITS <11:00> OF BOTH THE KERNEL
                        ;*      AND USER PAGE ADDRESS REGISTERS.  A "0" IS ROTATED THRU
                        ;*      THE REGISTERS FROM LEFT TO RIGHT.  THIS CHECKS THE OPERATION
                        ;*      OF THE PAR/PDR ADDRESS MUX, THE KT MUX, AND THE PAR
                        ;*      OUTPUT DATA LINES.
                        ;*
                        ;;*****************************************************************
022224  000004         TST15:  SCOPE

022226  012700  172340  1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR IN R0
022232  012703  000010  2$:     MOV     #10,R3          ;SETUP R3 TO COUNT 8 PAR'S
022236  012737  022244  001110         MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022244  005010         3$:     CLR     (R0)            ;CLEAR THE PAR
022246  011001                 MOV     (R0),R1         ;READ THE PAR INTO R1
022250  001401                 BEQ     4$              ;BRANCH IF PAR CLEARED OK
022252  104011                 ERROR   11              ;PAR WOULD NOT CLEAR
```

```
1848                                                                    ;FOR TIGHTER SCOPE LOOP
1849                                                                    ;REPLACE ERROR CALL WITH
1850                                                                    ;"BR 3$" = 000774
1851  022254  012704  167777          4$:     MOV     #167777,R4      ;LOAD "WALKING 0" TEST PATTERN IN R4
1852  022260  012737  022266  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1853  022266  005010                  5$:     CLR     (R0)            ;CLEAR THE PAR BEFORE LOADING DATA
1854  022270  010401                          MOV     R4,R1           ;LOAD DATA INTO R1
1855  022272  042701  170000                  BIC     #170000,R1      ;MASK UNUSED BITS OUT OF THE DATA
1856  022276  050110                          BIS     R1,(R0)         ;BIT SET THE TEST PATTERN INTO THE PAR
1857  022300  011002                          MOV     (R0),R2         ;READ THE PAR INTO R2
1858  022302  020102                          CMP     R1,R2           ;DOES DATA WRITTEN=DATA READ?
1859  022304  001401                          BEQ     6$              ;BRANCH IF YES
1860  022306  104012                          ERROR   12              ;PAR BITS DID NOT SET CORRECTLY
1861                                                                  ;FOR TIGHTER SCOPE LOOP
1862                                                                  ;REPLACE ERROR CALL WITH
1863                                                                  ;"BR 5$" = 000767
1864  022310  000261                  6$:     SEC                     ;SET THE C-BIT FOR THE ROTATE INST.
1865  022312  006004                          ROR     R4              ;ROTATE THE TEST PATTERN IN R4
1866  022314  103764                          BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
1867  022316  062700  000002                  ADD     #2,R0           ;GET NEXT PAR ADDRESS IN R0
1868  022322  077330                          SOB     R3,3$           ;BRANCH BACK UNTIL ALL PAR'S TESTED
1869  022324  022700  177660                  CMP     #UIPAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
1870  022330  103003                          BHIS    7$              ;BRANCH IF YES
1871  022332  012700  177640                  MOV     #UIPAR0,R0      ;LOAD FIRST USER PAR ADDR. IN R0
1872  022336  000735                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
1873  022340  012737  022226  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP OR ERROR POINTER TO 1$.
1874                                                                  ;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
```

```
                        ;;*****************************************************************
                        ;*TEST 16       BIT TEST OF KERNEL & USER PDR'S
                        ;*
                        ;*      THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
                        ;*      KERNEL AND USER PAGE DESCRIPTOR REGISTERS.  A "0" IS ROTATED
                        ;*      THRU THE REGISTERS FROM LEFT TO RIGHT.  SOME TEST PATTERNS WILL
                        ;*      BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
                        ;*      THE PAR/PDR ADDRESS MUX, KTMUX, AND PDR OUTPUT DATA LINES
                        ;*      ARE BEING CHECKED.
                        ;*
                        ;;*****************************************************************
022346  000004         TST16:  SCOPE

022350  012700  172300  1$:     MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST PDR IN R0
022354  012703  000010  2$:     MOV     #10,R3          ;SETUP R3 TO COUNT 8 PDR'S
022360  012737  022366  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022366  005010          3$:     CLR     (R0)            ;CLEAR THE PDR
022370  011001                  MOV     (R0),R1         ;READ THE PDR INTO R1
022372  001401                  BEQ     4$              ;BRANCH IF PDR CLEARED OK
022374  104011                  ERROR   11              ;PDR WOULD NOT CLEAR
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 3$" = 000774
022376  012704  077777  4$:     MOV     #077777,R4      ;LOAD "WALKING "0" TEST PATTERN IN R4
022402  012737  022410  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
022410  005010          5$:     CLR     (R0)            ;CLEAR THE PDR BEFORE LOADING DATA
022412  010401                  MOV     R4,R1           ;LOAD DATA INTO R1
022414  042701  100361          BIC     #100361,R1      ;MASK UNUSED BITS OUT OF THE DATA
022420  050110                  BIS     R1,(R0)         ;BIT SET THE TEST PATTERN INTO THE PDR
022422  011002                  MOV     (R0),R2         ;READ THE PDR INTO R2
022424  020102                  CMP     R1,R2           ;DOES DATA WRITTEN=DATA READ?
022426  001401                  BEQ     6$              ;BRANCH IF YES
022430  104012                  ERROR   12              ;PDR BITS DID NOT SET CORRECTLY
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 5$" = 000767
022432  000261          6$:     SEC                     ;SET THE C-BIT FOR THE ROTATE INST.
022434  006004                  ROR     R4              ;ROTATE THE TEST PATTERN IN R4
022436  103764                  BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
022440  062700  000002          ADD     #2,R0           ;GET NEXT PDR ADDRESS IN R0
022444  077330                  SOB     R3,3$           ;BRANCH BACK UNTIL ALL PDR'S TESTED
022446  022700  177620          CMP     #UIPDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
022452  103003                  BHIS    7$              ;BRANCH IF YES
022454  012700  177600          MOV     #UIPDR0,R0      ;LOAD FIRST USER PDR ADDR. IN R0
022460  000735                  BR      2$              ;BRANCH BACK TO TEST USER PDR'S
022462  012737  022350  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
                                                        ;LEAVE TEST WITH ALL WRITEABLE BITS IN
                                                        ;ALL PDR'S = 1

                        ;;*****************************************************************
                        ;*TEST 17       TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
                        ;*
                        ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
                        ;*      PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
                        ;*      THE OTHER.  THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
                        ;*      ADDR. MUX AND THE ADDR. DETECTION LOGIC.
```

```
1931                                    ;*
1932                                    ;;**********************************************************************
1933  022470  000004                    TST17:  SCOPE
```

```
1934
1935  022472  012700  172340          1$:   MOV   #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR INTO R0
1936  022476  012737  022510  001110  2$:   MOV   #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1937  022504  012703  000010          MOV   #10,R3               ;LOAD LOOP COUNTER TO DO 8 PAR'S
1938  022510  012701  177777          3$:   MOV   #-1,R1         ;LOAD TEST PATTERN INTO R1
1939  022514  005010                  CLR   (R0)                 ;CLEAR THE PAR
```

```
1940  022516  110110                          MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1941  022520  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1942  022522  042701  177400                  BIC     #177400,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1943  022526  020102                          CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO
1944  022530  001401                          BEQ     4$              ;BRANCH IF YES
1945  022532  104015                          ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1946                                                                  ;FOR TIGHTER SCOPE LOOP
1947                                                                  ;REPLACE ERROR CALL WITH
1948                                                                  ;"BR 3$" = 000766
1949  022534  012737  022542  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1950  022542  005010                  5$:     CLR     (R0)            ;CLEAR THE PAR
1951  022544  012701  177777                  MOV     #-1,R1          ;LOAD TEST, PATTERN INTO R1
1952  022550  110160  000001                  MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1953  022554  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1954  022556  042701  170377                  BIC     #170377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
1955  022562  020102                          CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1956  022564  001401                          BEQ     6$              ;BRANCH IF YES
1957  022566  104015                          ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1958                                                                  ;FOR TIGHTER SCOPE LOOP
1959                                                                  ;REPLACE ERROR CALL WITH
1960                                                                  ;"BR 5" = 000765
1961  022570  062700  000002          6$:     ADD     #2,R0           ;PUT ADDRESS OF NEXT PAR IN R0
1962  022574  077333                          SOB     R3,3$           ;BRANCH BACK UNTIL 8 PAR'S TESTED
1963  022576  022700  177660                  CMP     #UIPAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
1964  022602  103003                          BHIS    7$              ;BRANCH IF YES
1965  022604  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
1966  022610  000732                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
1967  022612  012737  022472  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1968
```

```
                                ;;*****************************************************************
                                ;*TEST 20       TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
                                ;*
                                ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
                                ;*      PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
                                ;*      THE OTHER.  THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
                                ;*      ADDR. MUX AND THE ADDR. DETECTION LOGIC.
                                ;*
                                ;;*****************************************************************
022620  000004                  TST20:  SCOPE
022622  012700  172300          1$:     MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST PDR INTO R0
022626  012737  022640  001110  2$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022634  012703  000010          3$:     MOV     #10,R3          ;LOAD LOOP COUNTER TO DO 8 PDR'S
022640  012701  177777          3$:     MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022644  005010                  3$:     CLR     (R0)            ;CLEAR THE PDR
022646  110110                  3$:     MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PDR
022650  011002                  3$:     MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022652  042701  177761          3$:     BIC     #177761,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
022656  020102                  3$:     CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO?
022660  001401                  3$:     BEQ     4$              ;BRANCH IF YES
022662  104015                  3$:     ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000766
022664  012737  022672  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
022672  005010                  5$:     CLR     (R0)            ;CLEAR THE PDR
022674  012701  177777          5$:     MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022700  110160  000001          5$:     MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
022704  011002                  5$:     MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022706  042701  100377          5$:     BIC     #100377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
022712  020102                  5$:     CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
022714  001401                  5$:     BEQ     6$              ;BRANCH IF YES
022716  104015                  5$:     ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000765
022720  062700  000002          6$:     ADD     #2,R0           ;PUT ADDRESS OF NEXT PDR IN R0
022724  077333                  6$:     SOB     R3,3$           ;BRANCH BACK UNTIL 8 PDR'S TESTED
022726  022700  177620          6$:     CMP     #UIPDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
022732  103003                  6$:     BHIS    7$              ;BRANCH IF YES
022734  012700  177600          6$:     MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
022740  000732                  6$:     BR      2$              ;BRANCH BACK TO TEST USER PDR'S
022742  012737  022622  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
```

```
                                ;;*****************************************************************
                                ;*TEST 21       PAR-PDR DUAL ADDRESSING TEST
                                ;*
                                ;*      THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
                                ;*      IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
                                ;*      SUBROUTINE AND THEN CLEARS JUST ONE OF THEM.  THE "CMPREG"
                                ;*      SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
                                ;*      THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
                                ;*      PAR OR PDR ADDRESS.  THE "CMPREG" SUBROUTINE REPORTS THE
                                ;*      ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
                                ;*      ANOTHER REGISTER WAS CLEARED.
                                ;*
                                ;*      THE PAR AND PDR CHIPS, PAR/PDR ADDR. MUX, AND ADDR. DETECTION
                                ;*      LOGIC ARE CHECKED.
                                ;*
                                ;;*****************************************************************
022750  000004                  TST21:  SCOPE

022752  012737  022770  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER 2$
022760  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
022764  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
022770  012706  001100          2$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
022774  004737  035520                  JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S IN PDR'S
023000  005010                          CLR     (R0)            ;CLEAR ONE OF THE KERNEL PDR'S
023002  004737  035612                  JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023006  062700  000002                  ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
023012  077312                          SOB     R3,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
023014  012737  023032  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
023022  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023026  012700  172340                  MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
023032  012706  001100          3$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023036  004737  035520                  JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
023042  005010                          CLR     (R0)            ;CLEAR ONE OF THE KERNEL PAR'S
023044  004737  035612                  JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023050  062700  000002                  ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
023054  077312                          SOB     R3,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
023056  012737  023074  001110          MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
023064  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023070  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
023074  012706  001100          4$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023100  004737  035520                  JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
023104  005010                          CLR     (R0)            ;CLEAR ONE OF THE USER PDR'S
023106  004737  035612                  JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023112  062700  000002                  ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PDR TO CLEAR
023116  077312                          SOB     R3,4$           ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
023120  012737  023136  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
023126  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023132  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
023136  012706  001100          5$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023142  004737  035520                  JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
023146  005010                          CLR     (R0)            ;CLEAR ONE OF THE USER PAR'S
023150  004737  035612                  JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
```

```
                                ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PAR TO CLEAR
                                SOB     R3,5$           ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
                                MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$

                        ;;*****************************************************************
                        ;*TEST 22       TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
                        ;*
                        ;*      THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
                        ;*      NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION.  THE
                        ;*      "SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
                        ;*      THE PAR/PDR'S.  THEN THEY ARE READ TO SEE THAT THEY REMAINED
                        ;*      UNCHANGED
                        ;*
                        ;;*****************************************************************
023170  000004         TST22:  SCOPE

023172  004737  035520  1$:     JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
023176  000005                 RESET                   ;ISSUE AN "INIT" BY EXECUTING A RESET
023200  012700  172300         MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
023204  012704  000010         MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
023210  011001         2$:     MOV     (R0),R1         ;READ A KERNEL PDR INTO R1
023212  022701  077416         CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
023216  001401                 BEQ     3$              ;BRANCH IF YES
023220  104055                 ERROR   55              ;KERNEL PDR AFFECTED BY A RESET
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 2$" = 000773
023222  062700  000002  3$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR
023226  077410                 SOB     R4,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
023230  012700  172340         MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
023234  012704  000010         MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
023240  011001         4$:     MOV     (R0),R1         ;READ A KERNEL PAR INTO R1
023242  022701  007777         CMP     #7777,R1        ;ARE ALL THE BITS STILL SET?
023246  001401                 BEQ     5$              ;BRANCH IF YES
023250  104055                 ERROR   55              ;KERNEL PAR AFFECTED BY A RESET
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 4$" = 000773
023252  062700  000002  5$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR
023256  077410                 SOB     R4,4$           ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
023260  012700  177600         MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
023264  012704  000010         MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
023270  011001         6$:     MOV     (R0),R1         ;READ A USER PDR INTO R1
023272  022701  077416         CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
023276  001401                 BEQ     7$              ;BRANCH IF YES
023300  104055                 ERROR   55              ;USER PDR AFFECTED BY A RESET
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 6$" = 000773
023302  062700  000002  7$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PDR
023306  077410                 SOB     R4,6$           ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED

023310  012700  177640         MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
023314  012704  000010         MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
023320  011001         8$:     MOV     (R0),R1         ;READ A USER PAR INTO R1
```

```
2126  023322  022701  007777          CMP     #7777,R1        ;ARE ALL THE BITS STILL SET?
2127  023326  001401                  BEQ     9$              ;BRANCH IF YES
2128  023330  104055                  ERROR   55              ;USER PAR AFFECTED BY A RESET
2129                                                          ;FOR TIGHTER SCOPE LOOP
2130                                                          ;REPLACE ERROR CALL WITH
2131                                                          ;"BR 8$" = 000773
2132  023332  062700  000002  9$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PAR
2133  023336  077410                  SOB     R4,8$           ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147                    ;*****************************************************************
2148                    ;*TEST 23       INSTRUCTION FETCH NOT RELOCATED IN MAINT. MODE
2149                    ;*
2150                    ;*      THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
2151                    ;*      MAINTENANCE MODE (DESTINATION-ONLY-RELOCATION), AN INSTRUCTION
2152                    ;*      FETCH IS NOT RELOCATED AND A RESET CLEARS THE MAINTENANCE BIT
2153                    ;*      (BIT 08) IN SR0.  IF THE "FETCH" IS RELOCATED, A PG.LENGTH ABORT
2154                    ;*      SHOULD OCCUR, CAUSING A HALT SINCE TRAP CATCHER IS PLACED IN VECTOR 250
2155                    ;*
2156                    ;*
2157                    ;*      NOTE:   A HALT MAY OCCUR IF MAINT. MODE NOT DISABLED BY RESET
2158                    ;*
2159                    ;*****************************************************************
2160  023340  000004  TST23:  SCOPE
2161  023342  004737  035432  1$:     JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2162  023346  012700  172300          MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
2163  023352  012704  000010          MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2164  023356  005020          2$:     CLR     (R0)+           ;CLEAR PDR - MAPPING PAGE NON-RES, 0 BLKS.
2165  023360  077402                  SOB     R4,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
2166  023362  012737  000252  000250  MOV     #MMVEC+2,MMVEC  ;LOAD TRAP CATCHER INTO MEM MGM. VECTOR
2167  023370  005037  000252          CLR     MMVEC+2         ;
2168                                                          ;A HALT WILL OCCUR IF RESET FAILS
2169  023374  012737  001006  172300  MOV     #1006,KIPDR0    ;TO DISABLE DEST.-ONLY RELOCATION
2170  023402  012737  023410  001110  MOV     #3$,$LPERR      ;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
2171  023410  012737  000400  177572  3$:     MOV     #BIT8,SR0       ;SET LOOP ON ERROR POINTER TO 3$
2172  023416  000005                  RESET                   ;TURN ON DEST-ONLY-RELOCATION
2173  023420  032737  000400  177572  BIT     #BIT8,SR0       ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
2174  023426  001403                  BEQ     4$              ;WAS MAINT. BIT (BIT 8) OF SR0 CLEARED?
2175  023430  005037  177572          CLR     SR0             ;BRANCH IF YES
2176  023434  104061                  ERROR   61              ;CLEAR SR0 SO ERROR CAN BE REPORTED
2177                                                          ;MAINT. MODE NOT DISABLED BY A RESET
2178                                                          ;FOR A TIGHTER SCOPE LOOP
2179                                                          ;REPLACE ERROR CALL WITH
2180  023436  012706  001100  4$:     MOV     #KERSTK,KSP     ;"BR 3$" = 000765
2181  023442  005037  177572          CLR     SR0             ;RESTORE STACK POINTER
                                                              ;BE SURE SR0 IS CLEAR
```

```
                                                MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. TRAP VECTOR
                                                MOV     #340,MMVEC+2    ;RESTORE MEM. MGMT VECTOR+2
                                                MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
                                                JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
```

```
;*********************************************************************
;*TEST 24       TEST THAT SOURCE NOT RELOCATED IN MAINTENANCE MODE
;*
;*      THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
;*      MAINTENANCE MODE, THE SOURCE IS NOT RELOCATED.  ONLY THE
;*      DESTINATION IS RELOCATED.  KERNEL PAR'S 3 & 4 ARE MAPPED TO
;*      PHYSICAL ADDRESS 60000-77776.  PDR4 IS SET TO ALLOW FULL READ/WRITE
;*      BUT PDR3 IS CLEAR ALLOWING TO ACCESS.  VIRTUAL ADDRESSES REFERENCING
;*      PAR/PDR'S 4 & 3 ARE USED IN AS DESTINATION AND SOURCE RESPECTIVELY.
;*      IF THE SOURCE IS RELOCATED IN MAINTENANCE A MEM. MGMT. TRAP WILL
;*      OCCUR AND THE ERROR WILL BE REPORTED.  KERNEL PG. 7 IS MAPPED R/W.
;*
;*********************************************************************
TST24:  SCOPE
1$:     JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
        MOV     #600,KIPAR3     ;MAP KERNAL PAGE 3 TO 12-16K
        MOV     #600,KIPAR4     ;MAP KERNAL PAGE 4 TO 12-16K
        MOV     #7600,KIPAR7    ;MAP KERNAL PAGE 7 TO THE I/O PAGE
        CLR     KIPDR3          ;MAP KERNAL PAGE 3 "NO ACCESS"
        MOV     #77406,KIPDR4   ;MAP KERNAL PAGE 4 R/W, 128 BLKS
        MOV     #77406,KIPDR7   ;MAP KERNAL PAGE 7 R/W, 128 BLKS
        MOV     #4$,MMVEC       ;SET M.M. VECTOR TO 4$ IN CASE OF ABORT
        MOV     #377,@#60000    ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC.
        MOV     #-1,R0          ;LOAD EXPECTED CONTENTS OF TEST LOC. IN R0
2$:     BIS     #BIT8,SR0       ;TURN ON DEST.-ONLY-RELOCATION
        MOVB    @#60000,@#100001;LOAD HI BYTE OF TEST LOC.-ABORT IF SOURCE RELOCATED
        RESET                   ;TURN OFF DEST.-ONLY-RELOCATION
        MOV     @#60000,R1      ;READ CONTENTS OF TEST LOC.
        CMP     R0,R1           ;WAS TEST LOCATION LOADED PROPERLY?
        BEQ     3$              ;BRANCH IF YES
        ERROR   62              ;TEST LOC. NOT LOADED - INST. WAS ABORTED
                                ;WHEN SOURCE WAS RELOCATED
                                ;FOR TIGHTER SCOPE LOOP REPLACE
                                ;ERROR CALL WITH
                                ;"BR 2$" = 000764
3$:     MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. VECTOR
        MOV     #77406,KIPDR3   ;MAP KERNAL PAGE 3 R/W, 128 BLKS
        MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
```

```
2238  023644  004737  035466          JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2239  023650  000430                  BR      TST25           ;;SKIP TO NEXT TEST
2240                                ;*  IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 4$
2241
2242  023652  042737  000400  177572  4$:     BIC     #BIT8,SR0       ;TURN OFF DEST.-ONLY-RELOCATION THRU PAR/PDR 7
2243  023660  013737  177572  001362          MOV     SR0,WASSR0      ;SAVE REST OF SR0 CONTENTS
2244  023666  013737  177576  001364          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2
2245  023674  010637  001354                  MOV     SP,WASR6        ;SAVE VALUE OF THE STACK POINTER
2246  023700  012637  001356                  MOV     (SP)+,TRAPPC    ;SAVE PC OF TRAP
2247  023704  012637  001360                  MOV     (SP)+,TRAPPS    ;SAVE PSW OF TRAP
2248  023710  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR1
2249  023716  104063                  ERROR   63              ;SOURCE APPARENTLY RELOCATED IN MAINT. MODE
2250                                                          ;FOR TIGHTER SCOPE LOOP
2251                                                          ;REPLACE ERROR CALL WITH A
2252                                                          ;"NOP" = 000240
2253  023720  013746  001360          MOV     TRAPPS,-(SP)    ;PUT PSW OF TRAP BACK ON THE STACK
2254  023724  013746  001356          MOV     TRAPPC,-(SP)    ;PUT PC OF TRAP BACK ON THE STACK
2255  023730  000002                  RTI                     ;RETURN TO TEST
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269                                ;;*****************************************************************
2270                                ;*TEST 25       RELOCATION & ADDER TEST (NO CARRIES)
2271                                ;*
2272                                ;*      THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
2273                                ;*      FOR THE REST OF THE PROGRAM.  IT THEN USES DIFFERENT
2274                                ;*      VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
2275                                ;*      TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE THREE
2276                                ;*      MEMORY MANAGEMENT ADDER CHIPS.  THE VALUES ARE SUCH
2277                                ;*      THAT NO CARRIES ARE GENERATED OUT OF ANY OF THE ADDER CHIPS.
2278                                ;*
2279                                ;*      THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
2280                                ;*      IS FORMED BY THE ADDERS IS TO WRITE A PATTERN TO VIRTUAL
2281                                ;*      LOCATION WITH MEMORY MGMT. IN THE MAINTENANCE MODE, AND
2282                                ;*      THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
2283                                ;*      HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
2284                                ;*
2285                                ;;*****************************************************************
2286  023732  000004                  TST25:  SCOPE
2287
2288  023734  012700  172340          1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2289  023740  005001                          CLR     R1              ;CLEAR R1
2290  023742  012702  000007                  MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
2291  023746  010120                  2$:     MOV     R1,(R0)+        ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
2292  023750  062701  000200                  ADD     #200,R1
2293  023754  077204                          SOB     R2,2$           ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
```

```
2294  023756  012710  007600                    MOV     #7600,(R0)      ;MAP KIPAR7 TO THE I/O PAGE
2295  023762  012700  172300                    MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2296  023766  012701  077406                    MOV     #77406,R1       ;LOAD PDR DATA INTO R1
2297  023772  012702  000010                    MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
2298  023776  010120                    3$:     MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2299  024000  077202                            SOB     R2,3$           ;       EXPANDABLE, READ/WRITE
2300
2301  024002  012737  024002  001110    4$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2302  024010  012700  067776                    MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2303  024014  012701  107776                    MOV     #107776,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2304  024020  012702  125250                    MOV     #125250,R2      ;LOAD TEST PATTERN INTO R2
2305  024024  012704  000600                    MOV     #600,R4 ;LOAD R4 WITH PAR VALUE
2306  024030  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2307  024034  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2308  024040  052737  000400  177572            BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2309  024046  010211                            MOV     R2,(R1)         ;LOAD 125250 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2310  024050  011003                            MOV     (R0),R3         ;READ 125250 BACK WITHOUT USING MEM. MGMT.
2311  024052  000005                            RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2312  024054  013710  001176                    MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2313  024060  020203                            CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2314                                                                    ;WRITTEN USING "DEST-ONLY-RELOC."?
2315  024062  001405                            BEQ     5$              ;BRANCH IF YES
2316  024064  010137  001376                    MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2317  024070  004737  036004                    JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2318  024074  104017                            ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2319                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2320                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2321                                                                    ;FORMED WRONG BY ADDERS USING
2322                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2323                                                                    ;FOR TIGHTER SCOPE LOOP
2324                                                                    ;REPLACE ERROR CALL WITH
2325                                                                    ;"BR 4$" = 000742
2326  024076                            5$:
2327  024076  012737  024076  001110    6$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2328  024104  012700  067776                    MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2329  024110  012701  102576                    MOV     #102576,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2330  024114  012702  125251                    MOV     #125251,R2      ;LOAD TEST PATTERN INTO R2
2331  024120  012704  000652                    MOV     #652,R4 ;LOAD R4 WITH PAR VALUE
2332  024124  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2333  024130  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2334  024134  052737  000400  177572            BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2335  024142  010211                            MOV     R2,(R1)         ;LOAD 125251 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2336  024144  011003                            MOV     (R0),R3         ;READ 125251 BACK WITHOUT USING MEM. MGMT.
2337  024146  000005                            RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2338  024150  013710  001176                    MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2339  024154  020203                            CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2340                                                                    ;WRITTEN USING "DEST-ONLY-RELOC."?
2341  024156  001405                            BEQ     7$              ;BRANCH IF YES
2342  024160  010137  001376                    MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2343  024164  004737  036004                    JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2344  024170  104017                            ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2345                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2346                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2347                                                                    ;FORMED WRONG BY ADDERS USING
2348                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2349                                                                    ;FOR TIGHTER SCOPE LOOP
```

```
2350                                                            ;REPLACE ERROR CALL WITH
2351                                                            ;"BR 6$" = 000742
2352  024172                                  7$:
2353  024172  012737  024172  001110  8$:     MOV     #8$,$LPERR    ;SET LOOP ON ERROR POINTER TO 8$
2354  024200  012700  067776                 MOV     #67776,R0     ;LOAD PHYSICAL ADDR. PBA INTO R0
2355  024204  012701  105276                 MOV     #105276,R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
2356  024210  012702  125252                 MOV     #125252,R2    ;LOAD TEST PATTERN INTO R2
2357  024214  012704  000625                 MOV     #625,R4 ;LOAD R4 WITH PAR VALUE
2358  024220  010437  172350                 MOV     R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2359  024224  011037  001176                 MOV     (R0),$TMP0    ;SAVE CONTENTS AT TEST LOCATION
2360  024230  052737  000400  177572         BIS     #BIT8,SR0     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2361  024236  010211                         MOV     R2,(R1)       ;LOAD 125252 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2362  024240  011003                         MOV     (R0),R3       ;READ 125252 BACK WITHOUT USING MEM. MGMT.
2363  024242  000005                         RESET                 ;TURN OFF MEMORY MGMT. MAINT. MODE
2364  024244  013710  001176                 MOV     $TMP0,(R0)    ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2365  024250  020203                         CMP     R2,R3         ;WAS SAME PATTERN READ BACK THAT WAS
2366                                                                ;WRITTEN USING "DEST-ONLY-RELOC."?
2367  024252  001405                         BEQ     9$            ;BRANCH IF YES
2368  024254  010137  001376                 MOV     R1,VIRT1      ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2369  024260  004737  036004                 JSR     PC,FORMPA     ;GO FORM PHYSICAL ADDRESS FOR TYPING
2370  024264  104017                         ERROR   17            ;TEST LOCATION DID NOT HAVE PATTERN
2371                                                                ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2372                                                                ;APPARENTLY PHYSICAL ADDR. WAS
2373                                                                ;FORMED WRONG BY ADDERS USING
2374                                                                ;THE VIRTUAL ADDR. AND KIPAR4
2375                                                                ;FOR TIGHTER SCOPE LOOP
2376                                                                ;REPLACE ERROR CALL WITH
2377                                                                ;"BR 8$" = 000742
2378  024266                                  9$:
2379
2380  024266  012737  024266  001110  10$:    MOV     #10$,$LPERR   ;SET LOOP ON ERROR POINTER TO 10$
2381  024274  012700  177776                 MOV     #PSW,R0       ;LOAD PHYS. ADDR. OF PSW INTO R0
2382  024300  012701  100076                 MOV     #100076,R1    ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2383  024304  012702  030340                 MOV     #030340,R2    ;LOAD DATA FOR PSW IN R2
2384  024310  012704  007777                 MOV     #7777,R4      ;LOAD R4 WITH PAR VALUE
2385  024314  010437  172350                 MOV     R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2386  024320  005010                         CLR     (R0)          ;CLEAR THE PSW
2387  024322  052737  000400  177572         BIS     #BIT8,SR0     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2388  024330  010211                         MOV     R2,(R1)       ;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2389  024332  011003                         MOV     (R0),R3       ;READ PSW BACK WITHOUT USING MEM. MGMT.
2390  024334  000005                         RESET                 ;TURN OFF MEM. MGMT MAINT. MODE
2391  024336  005010                         CLR     (R0)          ;CLEAR THE PSW
2392  024340  042703  000037                 BIC     #37,R3        ;MASK T-BIT & CC BITS OUT OF DATA READ
2393  024344  020203                         CMP     R2,R3         ;WAS PSW WRITTEN WHILE IN MAINT. MODE?
2394  024346  001405                         BEQ     11$           ;BRANCH IF YES
2395  024350  010137  001376                 MOV     R1,VIRT1      ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2396  024354  004737  036004                 JSR     PC,FORMPA     ;GO FORM PHYSICAL ADDR. FOR TYPING
2397  024360  104017                         ERROR   17            ;PSW DID NOT HAVE DATA THAT IT SHOULD HAVE,
2398                                                                ;APPARENTLY PHYS. ADDR. OF PSW WAS
2399                                                                ;NOT FORMED BY ADDERS USING THE
2400                                                                ;VIRTUAL ADDR. AND KIPAR4
2401                                                                ;FOR TIGHTER SCOPE LOOP
2402                                                                ;REPLACE ERROR CALL WITH
2403                                                                ;"BR 10$" = 000742
2404  024362  012737  024362  001110  11$:    MOV     #11$,$LPERR   ;SET LOOP ON ERROR POINTER TO 11$
2405  024370  012700  177776                 MOV     #PSW,R0       ;LOAD PHYS. ADDR. OF PSW INTO R0
```

```
2406  024374  012701  117776          MOV     #117776,R1      ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2407  024400  012702  030240          MOV     #030240,R2      ;LOAD DATA FOR PSW IN R2
2408  024404  012704  007600          MOV     #7600,R4        ;LOAD R4 WITH PAR VALUE
2409  024410  010437  172350          MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2410  024414  052737  000400  177572  BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2411  024422  010211                  MOV     R2,(R1)         ;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT. ADDR.)
2412  024424  011003                  MOV     (R0),R3         ;READ PSW BACK WITHOUT USING MEM. MGMT.
2413  024426  000005                  RESET                   ;TURN OFF MEM. MGMT. MAINT. MODE
2414  024430  005010                  CLR     (R0)            ;CLEAR THE PSW
2415  024432  042703  000037          BIC     #37,R3          ;MASK T-BIT & CC BITS OUT OF DATA READ
2416  024436  020203                  CMP     R2,R3           ;WAS PSW WRITTEN WHILE IN MAINT. MODE?
2417  024440  001405                  BEQ     12$             ;BRANCH IF YES
2418  024442  010137  001376          MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
2419  024446  004737  036004          JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDR. FOR TYPING
2420  024452  104017                  ERROR   17              ;PSW DID NOT HAVE DATA THAT IT SHOULD
2421                                                          ;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2422                                                          ;NOT FORMED BY ADDERS USING THE
2423                                                          ;VIRTUAL ADDR. AND KIPAR4
2424                                                          ;FOR TIGHTER SCOPE LOOP
2425                                                          ;REPLACE ERROR CALL WITH
2426                                                          ;"BR 11$" = 000743
2427  024454  012737  023734  001110  12$:    MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2428
2429                          ;;*****************************************************************
2430                          ;*TEST 26       RELOCATION & ADDER TEST (WITH CARRIES)
2431                          ;*
2432                          ;*      THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2433                          ;*      TEST TO VERIFY MEMORY MANAGEMENTS ABILITY TO CONSTRUCT
2434                          ;*      PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2435                          ;*      CONTENTS OF A PAGE ADDRESS REGISTER.  HOWEVER, THE VALUES
2436                          ;*      AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES FROM
2437                          ;*      CHIP TO CHIP AND CHECK "WRAPAROUND" TO ADDRESS 000000 BY
2438                          ;*      USING VIRTUAL ADDR. 100100 AND KIPAR4 = 7777.
2439                          ;*
2440                          ;;*****************************************************************
2441  024462  000004          TST26:  SCOPE
2442
2443  024464                  1$:                             ;KERNEL PAR'S AND PDR'S HAVE BEEN
2444                                                          ;SETUP BY THE PREVIOUS TEST
2445  024464  012737  024464  001110  2$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2446  024472  012700  064276          MOV     #64276,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2447  024476  012701  102176          MOV     #102176,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2448  024502  012702  125253          MOV     #125253,R2      ;LOAD TEST PATTERN INTO R2
2449  024506  012704  000621          MOV     #621,R4 ;LOAD R4 WITH PAR VALUE
2450  024512  010437  172350          MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2451  024516  011037  001176          MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2452  024522  052737  000400  177572  BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2453  024530  010211                  MOV     R2,(R1)         ;LOAD 125253 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2454  024532  011003                  MOV     (R0),R3         ;READ 125253 BACK WITHOUT USING MEM. MGMT.
2455  024534  000005                  RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2456  024536  013710  001176          MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2457  024542  020203                  CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2458                                                          ;WRITTEN USING "DEST-ONLY-RELOC."?
2459  024544  001405                  BEQ     3$              ;BRANCH IF YES
2460  024546  010137  001376          MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2461  024552  004737  036004          JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
```

```
2462  024556  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2463                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2464                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2465                                                                  ;FORMED WRONG BY ADDERS USING
2466                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2467                                                                  ;FOR TIGHTER SCOPE LOOP
2468                                                                  ;REPLACE ERROR CALL WITH
2469                                                                  ;"BR 2$" = 000742
2470  024560                          3$:
2471  024560  012737  024560  001110  4$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2472  024566  012700  064476                  MOV     #64476,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2473  024572  012701  112376                  MOV     #112376,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2474  024576  012702  125254                  MOV     #125254,R2      ;LOAD TEST PATTERN INTO R2
2475  024602  012704  000521                  MOV     #521,R4 ;LOAD R4 WITH PAR VALUE
2476  024606  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2477  024612  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2478  024616  052737  000400  177572          BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2479  024624  010211                          MOV     R2,(R1)         ;LOAD 125254 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2480  024626  011003                          MOV     (R0),R3         ;READ 125254 BACK WITHOUT USING MEM. MGMT.
2481  024630  000005                          RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2482  024632  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2483  024636  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2484                                                                  ;WRITTEN USING "DEST-ONLY-RELOC."?
2485  024640  001405                          BEQ     5$              ;BRANCH IF YES
2486  024642  010137  001376                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2487  024646  004737  036004                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2488  024652  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2489                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2490                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2491                                                                  ;FORMED WRONG BY ADDERS USING
2492                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2493                                                                  ;FOR TIGHTER SCOPE LOOP
2494                                                                  ;REPLACE ERROR CALL WITH
2495                                                                  ;"BR 4$" = 000742
2496  024654                          5$:
2497  024654  012737  024654  001110  6$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2498  024662  012700  061076                  MOV     #61076,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2499  024666  012701  106776                  MOV     #106776,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2500  024672  012702  125255                  MOV     #125255,R2      ;LOAD TEST PATTERN INTO R2
2501  024676  012704  000521                  MOV     #521,R4 ;LOAD R4 WITH PAR VALUE
2502  024702  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2503  024706  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2504  024712  052737  000400  177572          BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2505  024720  010211                          MOV     R2,(R1)         ;LOAD 125255 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2506  024722  011003                          MOV     (R0),R3         ;READ 125255 BACK WITHOUT USING MEM. MGMT.
2507  024724  000005                          RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2508  024726  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2509  024732  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2510                                                                  ;WRITTEN USING "DEST-ONLY-RELOC."?
2511  024734  001405                          BEQ     7$              ;BRANCH IF YES
2512  024736  010137  001376                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2513  024742  004737  036004                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2514  024746  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2515                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2516                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2517                                                                  ;FORMED WRONG BY ADDERS USING
```

```
2518                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2519                                                                    ;FOR TIGHTER SCOPE LOOP
2520                                                                    ;REPLACE ERROR CALL WITH
2521                                                                    ;"BR 6$" = 000742
2522  024750                                    7$:
2523  024750  012737  024750  001110  8$:      MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
2524  024756  012700  060076                   MOV     #60076,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2525  024762  012701  107776                   MOV     #107776,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2526  024766  012702  125256                   MOV     #125256,R2      ;LOAD TEST PATTERN INTO R2
2527  024772  012704  000501                   MOV     #501,R4 ;LOAD R4 WITH PAR VALUE
2528  024776  010437  172350                   MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2529  025002  011037  001176                   MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2530  025006  052737  000400  177572           BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2531  025014  010211                           MOV     R2,(R1)         ;LOAD 125256 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2532  025016  011003                           MOV     (R0),R3         ;READ 125256 BACK WITHOUT USING MEM. MGMT.
2533  025020  000005                           RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2534  025022  013710  001176                   MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2535  025026  020203                           CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2536                                                                    ;WRITTEN USING "DEST-ONLY-RELOC."?
2537  025030  001405                           BEQ     9$              ;BRANCH IF YES
2538  025032  010137  001376                   MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2539  025036  004737  036004                   JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2540  025042  104017                           ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2541                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2542                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2543                                                                    ;FORMED WRONG BY ADDERS USING
2544                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2545                                                                    ;FOR TIGHTER SCOPE LOOP
2546                                                                    ;REPLACE ERROR CALL WITH
2547                                                                    ;"BR 8$" = 000742
2548  025044                                    9$:
2549  025044  012737  025044  001110  10$:     MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2550  025052  012700  000000                   MOV     #000000,R0      ;LOAD PHYSICAL ADDR. PBA INTO R0
2551  025056  012701  100100                   MOV     #100100,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2552  025062  012702  125257                   MOV     #125257,R2      ;LOAD TEST PATTERN INTO R2
2553  025066  012704  007777                   MOV     #7777,R4        ;LOAD R4 WITH PAR VALUE
2554  025072  010437  172350                   MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2555  025076  011037  001176                   MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2556  025102  052737  000400  177572           BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2557  025110  010211                           MOV     R2,(R1)         ;LOAD 125257 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2558  025112  011003                           MOV     (R0),R3         ;READ 125257 BACK WITHOUT USING MEM. MGMT.
2559  025114  000005                           RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2560  025116  013710  001176                   MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2561  025122  020203                           CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2562                                                                    ;WRITTEN USING "DEST-ONLY-RELOC."?
2563  025124  001405                           BEQ     11$             ;BRANCH IF YES
2564  025126  010137  001376                   MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2565  025132  004737  036004                   JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2566  025136  104017                           ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2567                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2568                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2569                                                                    ;FORMED WRONG BY ADDERS USING
2570                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2571                                                                    ;FOR TIGHTER SCOPE LOOP
2572                                                                    ;REPLACE ERROR CALL WITH
2573                                                                    ;"BR 10$" = 000742
```

```
                                        11$:
2574  025140                                    
2575  025140  012737  024464  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2576                                    
2577                                    ;;*****************************************************************
2578                                    ;*TEST 27       READ AND WRITE WHILE IN RELOCATE MODE
2579                                    ;*
2580                                    ;*      THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2581                                    ;*      READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2582                                    ;*      060000-067600.  ONE LOCATION IN EVERY BLOCK (32. WORDS)
2583                                    ;*      IS WRITTEN USING PAR4 AND READ USING PAR5.  THIS IS
2584                                    ;*      DONE IN BOTH USER AND KERNEL MODES.  THE USER PAR/PDR'S
2585                                    ;*      ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2586                                    ;*      MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2587                                    ;*      OF THE PROGRAM.  THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
2588                                    ;*      IS CHECKED BY READING AND WRITING IN USER MODE.  REMEMBER
2589                                    ;*      ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2590                                    ;*      MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2591                                    ;*      THE PAR/PDR'S TO EXECUTE.
2592                                    ;*
2593                                    ;*      WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
2594                                    ;*      NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2595                                    ;*      PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
2596                                    ;*      USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2597                                    ;*      USER MODE).  IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2598                                    ;*      TO 8$ WHERE THE TRAP IS REPORTED.
2599                                    ;*
2600                                    ;;*****************************************************************
2601  025146  000004                    1ST27:  SCOPE
2602                                    
2603  025150  005037  177776            1$:     CLR     PSW             ;START IN KERNEL MODE
2604  025154  012704  000577                    MOV     #577,R4         ;LOAD R4 WITH VALUE FOR PAR4
2605  025160  012705  000600                    MOV     #600,R5         ;LOAD R5 WITH VALUE FOR PAR5
2606  025164  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR4
2607  025170  010537  172352                    MOV     R5,KIPAR5       ;LOAD KERNEL PAR5
2608  025174  012700  177640                    MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2609  025200  005001                            CLR     R1              ;CLEAR R1
2610  025202  012702  000007                    MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
2611  025206  010120                    2$:     MOV     R1,(R0)+        ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2612  025210  062701  000200                    ADD     #200,R1
2613  025214  077204                            SOB     R2,2$           ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2614  025216  012710  007600                    MOV     #7600,(R0)      ;MAP USER PAR7 TO THE I/O PAGE
2615  025222  012700  177600                    MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2616  025226  012701  077406                    MOV     #77406,R1       ;LOAD PDR DATA INTO R1
2617  025232  012702  000010                    MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
2618  025236  010120                    3$:     MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2619  025240  077202                            SOB     R2,3$           ;       EXPANDABLE, READ/WRITE
2620  025242  105037  177610                    CLRB    UIPDR4          ;MAP USER SPACE NON-RESIDENT WHILE
2621  025246  105037  177612                    CLRB    UIPDR5          ;       TESTING KERNEL SPACE
2622  025252  010537  177650                    MOV     R5,UIPAR4       ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2623  025256  010437  177652                    MOV     R4,UIPAR5
2624  025262  012737  000001  177572          MOV     #1,SR0          ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
2625  025270  012737  025322  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2626  025276  012737  025534  000250          MOV     #8$,MMVEC       ;SET M. M. TRAP VECTOR TO 8$
2627  025304  013737  177776  001176  4$:     MOV     PSW,$TMPO       ;SAVE PSW IN CASE OF ERROR
2628  025312  012700  100100                    MOV     #100100,R0      ;PUT VIRTUAL ADDR. THAT USER PAR4 IN R0
2629  025316  012701  120000                    MOV     #120000,R1      ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
```

```
2630  025322  010010                  5$:     MOV     R0,(R0)         ;WRITE TO TEST LOC. USING PAR4
2631  025324  011102                          MOV     (R1),R2         ;READ THE SAME LOC., BUT USING PAR5
2632  025326  020002                          CMP     R0,R2           ;DID WE READ WHAT WE WROTE?
2633  025330  001411                          BEQ     6$              ;BRANCH IF YES
2634  025332  010137  001400                  MOV     R1,VIRT2        ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2635  025336  010037  001376                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2636  025342  004737  036004                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS  BEING USED
2637  025346  104020                          ERROR   20              ;READING LOC. USING PAR5 AND A VIRT.
2638                                                                  ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2639                                                                  ;PAR4 AND VIRT. ADDRESS.
2640                                                                  ;FOR TIGHTER SCOPE LOOP
2641                                                                  ;REPLACE ERROR CALL WITH
2642                                                                  ;"BR 5$" = 000765
2643  025350  013700  001376                  MOV     VIRT1,R0        ;RESTORE VBA IN R0
2644  025354  062700  000100          6$:     ADD     #100,R0         ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
2645  025360  062701  000100                  ADD     #100,R1
2646  025364  020127  127700                  CMP     R1,#127700      ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
2647  025370  001354                          BNE     5$              ;BRANCH IF NO
2648  025372  032737  140000  177776          BIT     #140000,PSW     ;HAVE WE DONE TEST IN USER MODE YET?
2649  025400  001026                          BNE     7$              ;BRANCH IF YES
2650  025402  010437  177650                  MOV     R4,UIPAR4       ;LOAD USER PAR4
2651  025406  010537  177652                  MOV     R5,UIPAR5       ;LOAD USER PAR5
2652  025412  112737  000006  177610          MOVB    #6,UIPDR4       ;MAP USER SPACE R/W TO TEST IT
2653  025420  112737  000006  177612          MOVB    #6,UIPDR5
2654  025426  105037  172310                  CLRB    KIPDR4          ;MAP  KERNEL SPACE NON-RESIDENT WHILE
2655  025432  105037  172312                  CLRB    KIPDR5          ;       TESTING USER SPACE
2656  025436  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2657  025442  010437  172352                  MOV     R4,KIPAR5
2658  025446  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
2659  025454  000713                          BR      4$              ;GO BACK AND READ/WRITE IN USER MODE
2660  025456  005037  177776          7$:     CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
2661  025462  012737  077406  172310          MOV     #77406,KIPDR4   ;REMAP KERNEL PAGES READ/WRITE
2662  025470  012737  077406  172312          MOV     #77406,KIPDR5
2663  025476  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL  AND USER PAR'S 4 & 5
2664  025502  010537  172352                  MOV     R5,KIPAR5       ;  BACK TO 12-16K
2665  025506  010537  177650                  MOV     R5,UIPAR4
2666  025512  010537  177652                  MOV     R5,UIPAR5
2667  025516  012737  002314  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2668  025524  012737  025150  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2669  025532  000427                          BR      TST30           ;;GO TO NEXT TEST
2670
2671  025534  012637  001356          8$:     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
2672  025540  012637  001360                  MOV     (KSP)+,TRAPPS
2673                                                                  ;PROGRAM WILL TRAP TO HERE IF TRY
2674                                                                  ;TO USE USER PDR'S WHEN IN KERNEL MODE
2675                                                                  ;OR KERNEL PDR'S WHEN IN USER MODE
2676  025544  010037  001376                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2677  025550  004737  036004                  JSR     PC,FORMPA       ;GO FORM THE PHYSICAL ADDRESS  BEING USED
2678  025554  013737  177572  001362          MOV     SR0,WASSR0      ;SAVE SR0 & SR2  FOR ERROR REPORT
2679  025562  013737  177576  001364          MOV     SR2,WASSR2
2680  025570  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2681  025576  104052                          ERROR   52              ;M.M. TRAP WHILE IN RELOCATE MODE -
2682                                                                  ;REFERENCED WRONG SET OF PDR'S
2683                                                                  ;FOR TIGHTER SCOPE LOOP
2684                                                                  ;REPLACE ERROR CALL WITH
2685                                                                  ;A "NOP" = 000240
```

```
                                        MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
                                        MOV     TRAPPC,-(KSP)
                                        RTI                     ;RETURN TO TEST


                                ;;*****************************************************************
                                ;*TEST 30       W-BIT LOGIC TEST, KERNEL PDR'S
                                ;*
                                ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
                                ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
                                ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
                                ;*      77776,77776, & 777776 RESPECTIVELY).
                                ;*      WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
                                ;*      EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS.  THE PDR'S
                                ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
                                ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
                                ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  KERNEL PDR'S 3,4,5,6
                                ;*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
                                ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO.  THE
                                ;*      W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
                                ;*      ARE BEING CHECKED.
                                ;;*****************************************************************
                                TST30:  SCOPE
                                1$:
                                        JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
                                        MOV     #4,R2           ;SET LOOP COUNTER TO 4
                                        MOV     #KIPAR3,R0      ;LOAD ADDRESS OF PAR3 INTO R0
                                        MOV     #600,R1         ;LOAD "12-16K" PAR VALUE INTO R1
                                2$:     MOV     R1,(R0)+        ;MAP PARS 3-6 TO 12-16K
                                        SOB     R2,2$           ;LOOP TIL ALL 4 OF THEM LOADED
                                        MOV     #KIPDR0,R5      ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
                                        MOV     #10,R4          ;SET LOOP COUNTER TO 8
                                        MOV     #17776,R3       ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
                                        MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
                                3$:     MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
                                        MOV     #10,R2          ;SET LOOP COUNTER TO 8
                                        MOV     #77406,R1       ;PUT "W-BIT OFF DATA" INTO R1
                                4$:     MOV     R1,(R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
                                        SOB     R2,4$           ;LOOP UNTIL ALL OF THEM SETUP
                                        MOV     (R3),(R3)       ;DO "DATO" TO VIRTUAL ADDR.-SETTING A W-BIT
                                        BIT     (R5),#WBIT      ;DID THAT CAUSE W-BIT TO BE SET?
                                        BNE     5$              ;BRANCH IF YES
                                        ERROR   21              ;W-BIT DID NOT GET SET IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000763
                                        BR      8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
                                5$:     MOV     #10,R2          ;SET LOOP COUNTER TO 8
                                        MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
                                6$:     BIT     (R0),#WBIT      ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
                                        BEQ     7$              ;BRANCH IF YES
                                        CMP     R5,R0           ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
                                        BEQ     7$              ;BRANCH IF YES
                                        ERROR   22              ;W-BIT GOT SET IN MORE THAN ONE PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
```

```
2742                                                            ;"BR 3$" = 000750
2743  025742  062700  000002          7$:     ADD     #2,R0     ;POINT R0 TO NEXT PDR TO BE CHECKED
2744  025746  077211                          SOB     R2,6$     ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2745  025750  010115                          MOV     R1,(R5)   ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2746  025752  031527  000100                  BIT     (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2747  025756  001401                          BEQ     8$        ;BRANCH IF YES
2748  025760  104023                          ERROR   23        ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2749                                                            ;FOR TIGHTER SCOPE LOOP
2750                                                            ;REPLACE ERROR CALL WITH
2751                                                            ;"BR 3$" = 000740
2752  025762  062705  000002          8$:     ADD     #2,R5     ;POINT R5 TO THE NEXT PDR TO BE TESTED
2753  025766  062703  020000                  ADD     #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2754  025772  077445                          SOB     R4,3$     ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2755  025774  012737  025614  001110          MOV     #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2756  026002  004737  035466                  JSR     PC,TON    ;TURN T-BIT BACK ON FOR NEXT TEST
2757
2758                                  ;*****************************************************************
2759                                  ;*TEST 31       W-BIT LOGIC TEST, USER PDR'S
2760                                  ;*
2761                                  ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2762                                  ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2763                                  ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2764                                  ;*      77776,77776, & 777776 RESPECTIVELY).
2765                                  ;*      WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
2766                                  ;*      EIGHT (8) USER PAGE DESCRIPTOR REGISTERS.  THE PDR'S
2767                                  ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2768                                  ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2769                                  ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  USER PDR'S 3,4,5,6
2770                                  ;*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
2771                                  ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO.  THE
2772                                  ;*      W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
2773                                  ;*      ARE BEING CHECKED.
2774                                  ;*****************************************************************
2775  026006  000004                  TST31:  SCOPE
2776  026010  012737  140000  177776  1$:     MOV     #140000,PSW ;GO TO USER MODE FOR THIS TEST
2777  026016  004737  035432                  JSR     PC,TOFF   ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2778  026022  012702  000004                  MOV     #4,R2     ;SET LOOP COUNTER TO 4
2779  026026  012700  177646                  MOV     #UIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
2780  026032  012701  000600                  MOV     #600,R1   ;LOAD "12-16K" PAR VALUE INTO R1
2781  026036  010120                  2$:     MOV     R1,(R0)+  ;MAP PARS 3-6 TO 12-16K
2782  026040  077202                          SOB     R2,2$     ;LOOP TIL ALL 4 OF THEM LOADED
2783  026042  012705  177600                  MOV     #UIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2784  026046  012704  000010                  MOV     #10,R4    ;SET LOOP COUNTER TO 8
2785  026052  012703  017776                  MOV     #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2786  026056  012737  026064  001110          MOV     #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
2787  026064  012700  177600          3$:     MOV     #UIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2788  026070  012702  000010                  MOV     #10,R2    ;SET LOOP COUNTER TO 8
2789  026074  012701  077406                  MOV     #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
2790  026100  010120                  4$:     MOV     R1,(R0)+  ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2791  026102  077202                          SOB     R2,4$     ;LOOP UNTIL ALL OF THEM SETUP
2792  026104  011313                          MOV     (R3),(R3) ;DO "DATO" TO VIRTUAL ADDR.-SETTING A W-BIT
2793  026106  031527  000100                  BIT     (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
2794  026112  001002                          BNE     5$        ;BRANCH IF YES
2795  026114  104021                          ERROR   21        ;W-BIT DID NOT GET SET IN PDR
2796                                                            ;FOR TIGHTER SCOPE LOOP
2797                                                            ;REPLACE ERROR CALL WITH
```

```
2798
2799  026116  000422                        BR      8$              ;"BR 3$" = 000763
2800  026120  012702  000010          5$:   MOV     #10,R2          ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2801  026124  012700  177600                MOV     #UIPDR0,R0      ;SET LOOP COUNTER TO 8
2802  026130  031027  000100          6$:   BIT     (R0),#WBIT      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2803  026134  001403                        BEQ     7$              ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2804  026136  020500                        CMP     R5,R0           ;BRANCH IF YES
2805  026140  001401                        BEQ     7$              ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2806  026142  104022                        ERROR   22              ;BRANCH IF YES$
2807                                                                ;W-BIT GOT SET IN MORE THAN ONE PDR
2808                                                                ;FOR TIGHTER SCOPE LOOP
2809                                                                ;REPLACE ERROR CALL WITH
2810  026144  062700  000002          7$:   ADD     #2,R0           ;"BR 3$" = 000750
2811  026150  077211                        SOB     R2,6$           ;POINT R0 TO NEXT PDR TO BE CHECKED
2812  026152  010115                        MOV     R1,(R5)         ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2813  026154  031527  000100                BIT     (R5),#WBIT      ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2814  026160  001401                        BEQ     8$              ;DID WRITING PDR CLEAR THE W-BIT?
2815  026162  104023                        ERROR   23              ;BRANCH IF YES
2816                                                                ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2817                                                                ;FOR TIGHTER SCOPE LOOP
2818                                                                ;REPLACE ERROR CALL WITH
2819  026164  062705  000002          8$:   ADD     #2,R5           ;"BR 3$" = 000740
2820  026170  062703  020000                ADD     #20000,R3       ;POINT R5 TO THE NEXT PDR TO BE TESTED
2821  026174  077445                        SOB     R4,3$           ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2822  026176  012737  026010  001110        MOV     #1$,$LPERR      ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2823  026204  004737  035466                JSR     PC,TON          ;RESET LOOP ON ERROR POINTER TO 1$
2824  026210  005037  177776                CLR     PSW             ;TURN T-BIT BACK ON FOR NEXT TEST
2825                                                                ;BACK TO KERNEL MODE BEFORE LEAVING
2826                                ;;*****************************************************************
2827                                ;*TEST 32       TEST "W-BIT NOT SET" CASES
2828                                ;*
2829                                ;*      THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES
2830                                ;*      NOT GET SET ON A WRITE.  FIRST CASE IS THAT THE W-BIT
2831                                ;*      SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO
2832                                ;*      STATUS REG SR0 (KERNEL PDR 7 IS USED).  SECOND CASE IS THAT
2833                                ;*      THE W-BIT IS NOT SET IF THE "DATO" IS ABORTED DUE TO AN
2834                                ;*      ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED).
2835                                ;*
2836                                ;;*****************************************************************
2837  026214  000004                TST32:  SCOPE
2838
2839  026216  004737  035432          1$:   JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2840  026222  012701  077406                MOV     #77406,R1       ;PUT "W-BIT OFF" VALUE FOR PDR IN R1
2841  026226  012737  026234  001110        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2842  026234  010137  172316          2$:   MOV     R1,KIPDR7       ;LOAD KERNEL PDR 7 TO CLEAR W-BIT
2843  026240  013700  177572                MOV     SR0,R0          ;READ PRESENT CONTENTS OF STATUS REG. 0
2844  026244  010037  177572                MOV     R0,SR0          ;WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
2845  026250  013702  172316                MOV     KIPDR7,R2       ;READ CONTENTS OF KIPDR7 INTO R2
2846  026254  020102                        CMP     R1,R2           ;WAS W-BIT LEFT CLEARED?
2847  026256  001401                        BEQ     3$              ;BRANCH IF YES
2848  026260  104024                        ERROR   24              ;W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
2849                                                                ;FOR TIGHTER SCOPE LOOP
2850                                                                ;REPLACE ERROR CALL WITH
2851                                                                ;"BR 2$" = 000765
2852  026262  012737  026262  001110  3$:   MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2853  026270  010137  172306                MOV     R1,KIPDR3       ;LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
```

```
2854  026274  012737  026306  000004          MOV     #4$,ERRVEC      ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2855  026302  005237  060001                  INC     60001           ;CAUSE ODD ADDRESS ABORT THRU LOC. 4
2856  026306  012706  001100          4$:     MOV     #KERSTK,KSP     ;RESTORE THE STACK POINTER
2857  026312  013702  172306                  MOV     KIPDR3,R2       ;READ KIPDR3 INTO R2
2858  026316  020102                          CMP     R1,R2           ;WAS W-BIT LEFT CLEARED?
2859  026320  001401                          BEQ     5$              ;BRANCH IF YES
2860  026322  104025                          ERROR   25              ;W-BIT GOT SET DURING AN ODD ADDR. ABORT
2861                                                                  ;FOR TIGHTER SCOPE LOOP
2862                                                                  ;REPLACE ERROR CALL WITH
2863                                                                  ;"BR 3$" = 000757
2864  026324  012737  002246  000004  5$:     MOV     #TIMERR,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2865  026332  012737  026216  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2866  026340  004737  035466                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2867
2868                                  ;;*****************************************************************
2869                                  ;*
2870                                  ;*      THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2871                                  ;*      TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2872                                  ;*      ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2873                                  ;*      VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2874                                  ;*      THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2875                                  ;*      ARE CHECKED IN SR0.  SO THE SR0 AND SR2 LOGIC AND THE
2876                                  ;*      KT ERROR LOGIC ARE CHECKED.
2877                                  ;*
2878                                  ;;*****************************************************************
2879
2880                                  ;;*****************************************************************
2881                                  ;*TEST 33       NON-RESIDENT ABORT TEST (ACF=0&4)
2882                                  ;*
2883                                  ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2884                                  ;*      LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2885                                  ;*      USER MODES.  PDR 4 IS LOADED WITH ACF'S = 0&4 AND
2886                                  ;*      THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
2887                                  ;*
2888                                  ;;*****************************************************************
2889  026344  000004                  TST33:  SCOPE
2890
2891  026346  012700  000600          1$:     MOV     #600,R0         ;LOAD DATA FOR PAR'S INTO R0
2892  026352  010037  172346                  MOV     R0,KIPAR3       ;MAP KERNEL PAR'S 3&4 TO 12-16K
2893  026356  010037  172350                  MOV     R0,KIPAR4
2894  026362  010037  177646                  MOV     R0,UIPAR3       ;MAP USER PAR'S 3&4 TO 12-16K
2895  026366  010037  177650                  MOV     R0,UIPAR4
2896  026372  012737  077406  172306          MOV     #77406,KIPDR3   ;MAP KERNEL PDR 3  128 BLKS, READ-WRITE
2897  026400  012737  077406  177606          MOV     #77406,UIPDR3   ;MAP USER PDR 3  128 BLKS, READ-WRITE
2898  026406  012700  060000                  MOV     #60000,R0       ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2899  026412  012701  100000                  MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2900  026416  012703  100011                  MOV     #100011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., KERNEL, PG.4
2901  026422  012702  077400                  MOV     #77400,R2       ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2902  026426  012737  026470  000250  2$:     MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2903  026434  010237  172310                  MOV     R2,KIPDR4       ;LOAD ACF TEST VALUE INTO KIPDR4
2904  026440  010237  177610                  MOV     R2,UIPDR4       ;LOAD ACF TEST VALUE INTO UIPDR4
2905  026444  012737  026452  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2906  026452  005010                  3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
2907  026454  013737  177776  001176          MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
2908  026462  005211                  4$:     INC     (R1)            ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2909  026464  104026                          ERROR   26              ;MEM. MGMT. ABORT DID NOT OCCUR
```

```
2910                                                    ;FOR TIGHTER SCOPE LOOP
2911                                                    ;REPLACE ERROR CALL WITH
2912                                                    ;"BR 3$" = 000772
2913  026466  000425                   BR      8$       ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2914  026470  062706  000004    5$:    ADD     #4,SP    ;RESTORE STACK POINTER
2915  026474  005710                   TST     (R0)     ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2916  026476  001401                   BEQ     6$       ;BRANCH IF YES
2917  026500  104027                   ERROR   27       ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2918                                                    ;FOR TIGHTER SCOPE LOOP
2919                                                    ;REPLACE ERROR CALL WITH
2920                                                    ;"BR 3$" = 000764
2921  026502  013737  177572  001362  6$:    MOV     SR0,WASSR0  ;READ STATUS REGISTER 0
2922  026510  013737  177576  001364       MOV     SR2,WASSR2  ;READ STATUS REGISTER 2
2923  026516  020337  001362             CMP     R3,WASSR0   ;DID SR0 REPORT NON-RESIDENT ERROR CORRECTLY?
2924  026522  001401                   BEQ     7$       ;BRANCH IF YES
2925  026524  104030                   ERROR   30       ;SR0 DID NOT REPORT NON-RES. ERROR CORRECTLY
2926                                                    ;FOR TIGHTER SCOPE LOOP
2927                                                    ;REPLACE ERROR CALL WITH
2928                                                    ;"BR 3$" = 000752
2929  026526  012704  026462    7$:    MOV     #4$,R4   ;LOAD R4 WITH WHAT SR2 SHOULD READ
2930  026532  020437  001364             CMP     R4,WASSR2   ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2931  026536  001401                   BEQ     8$       ;BRANCH IF YES
2932  026540  104031                   ERROR   31       ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2933                                                    ;FOR TIGHTER SCOPE LOOP
2934                                                    ;REPLACE ERROR CALL WITH
2935                                                    ;"BR 3$" = 000744
2936  026542  042737  160000  177572  8$:    BIC     #160000,SR0 ;CLEAR THE ERROR BITS IN SR0
2937  026550  032737  140000  001176       BIT     #140000,$TMP0  ;HAS ACF=0&4 BEEN TESTED IN USER YET
2938  026556  001006                   BNE     9$       ;BRANCH IF YES
2939  026560  012703  100151             MOV     #100151,R3  ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., USER, PG.4
2940  026564  012737  140000  177776     MOV     #140000,PSW ;GO TO USER MODE
2941  026572  000715                   BR      2$       ;REPEAT TEST IN USER MODE
2942  026574  022702  077404    9$:    CMP     #77404,R2   ;HAS ACF=4 BEEN TESTED YET?
2943  026600  001407                   BEQ     10$      ;BRANCH IF YES
2944  026602  012702  077404             MOV     #77404,R2   ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2945  026606  012703  100011             MOV     #100011,R3  ;LOAD R3 WITH WHAT SR0 SHOULD READ-N.R.,KERNEL,PG. 4
2946  026612  005037  177776             CLR     PSW      ;GO BACK TO KERNEL MODE
2947  026616  000703                   BR      2$       ;GO BACK & TEST ACF=4 IN SAME MODE
2948  026620  005037  177776    10$:    CLR     PSW      ;GO BACK TO KERNEL MODE BEFORE LEAVING
2949  026624  012737  026346  001110     MOV     #1$,$LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
2950  026632  012737  002314  000250     MOV     #MGMERR,MMVEC  ;RESTORE ADDRESS OF NORMAL MEMORY
2951                                                    ;MANAGEMENT ERROR ROUTINE TO MMVEC
2952
2953                          ;;*****************************************************************
2954                          ;*TEST 34       READ-ONLY ABORT TEST (ACF=2)
2955                          ;*
2956                          ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2957                          ;*      LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2958                          ;*      USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
2959                          ;*      PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
2960                          ;*
2961                          ;;*****************************************************************
2962  026640  000004          TST34:  SCOPE
2963  026642                  1$:                      ;KERNEL & USER PAR'S 3 & 4 AND PDR 3
2964                                                   ;ARE SETUP FROM LAST TEST
2965  026642  012700  060000          MOV     #60000,R0  ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
```

```
2966  026646  012701  100000              MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2967  026652  012703  020011              MOV     #20011,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG.4
2968  026656  012702  077402              MOV     #77402,R2       ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
2969  026662  012737  026724  000250  2$: MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2970  026670  010237  172310              MOV     R2,KIPDR4       ;LOAD ACF=2 INTO KIPDR4
2971  026674  010237  177610              MOV     R2,UIPDR4       ;LOAD ACF=2 INTO UIPDR4
2972  026700  012737  026706  001110      MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2973  026706  005010              3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
2974  026710  013737  177776  001176      MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
2975  026716  005211              4$:     INC     (R1)            ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
2976  026720  104026                      ERROR   26              ;MEM. MGMT. ABORT DID NOT OCCUR
2977                                                              ;FOR TIGHTER SCOPE LOOP
2978                                                              ;REPLACE ERROR CALL WITH
2979                                                              ;"BR 3$" = 000772
2980  026722  000425                      BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2981  026724  062706  000004      5$:     ADD     #4,SP           ;RESTORE STACK POINTER
2982  026730  005710                      TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2983  026732  001401                      BEQ     6$              ;BRANCH IF YES
2984  026734  104027                      ERROR   27              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2985                                                              ;FOR TIGHTER SCOPE LOOP
2986                                                              ;REPLACE ERROR CALL WITH
2987                                                              ;"BR 3$" = 000764
2988  026736  013737  177572  001362  6$: MOV     SR0,WASSR0      ;READ STATUS REG. 0
2989  026744  013737  177576  001364      MOV     SR2,WASSR2      ;READ STATUS REG. 2
2990  026752  020337  001362              CMP     R3,WASSR0       ;DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
2991  026756  001401                      BEQ     7$              ;BRANCH IF YES
2992  026760  104030                      ERROR   30              ;SR0 DID NOT REPORT R/O ERROR CORRECTLY
2993                                                              ;FOR TIGHTER SCOPE LOOP
2994                                                              ;REPLACE ERROR CALL WITH
2995                                                              ;"BR 3$" = 000752
2996  026762  012704  026716      7$:     MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
2997  026766  020437  001364              CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2998  026772  001401                      BEQ     8$              ;BRANCH IF YES
2999  026774  104031                      ERROR   31              ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
3000                                                              ;FOR TIGHTER SCOPE LOOP
3001                                                              ;REPLACE ERROR CALL WITH
3002                                                              ;"BR 3$" = 000744
3003  026776  042737  160000  177572  8$: BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
3004  027004  032737  140000  001176      BIT     #140000,$TMP0   ;HAS ACF=2 BEEN TESTED IN USER MODE?
3005  027012  001006                      BNE     9$              ;BRANCH IF YES
3006  027014  012703  020151              MOV     #20151,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4
3007  027020  012737  140000  177776      MOV     #140000,PSW     ;GO TO USER MODE
3008  027026  000715                      BR      2$              ;REPEAT TEST IN USER MODE
3009  027030  005037  177776      9$:     CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
3010  027034  012737  026642  001110      MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3011  027042  012737  002314  000250      MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
3012                                                              ;MANAGEMENT ERROR ROUTINE TO MMVEC.
3013                                ;;*****************************************************************
3014                                ;*TEST 35       TEST ILLEGAL MODE "01"
3015                                ;*
3016                                ;*      THIS TEST CHECKS TO SEE THAT A 01 IN THE CURRENT MODE BITS OF THE
3017                                ;*      PSW WHILE MEMORY MANAGEMENT IS ON IS ILLEGAL.  A
3018                                ;*      MEMORY MANAGEMENT ABORT SHOULD OCCUR AND STATUS REGISTER 0
3019                                ;*      SHOULD REPORT NON-RESIDENT ABORT, MODE = 01, PAGE = 1 (100043).  STATUS
3020                                ;*      REGISTER 2 SHOULD LOCKUP THE ADDRESS OF THE INSTRUCTION
3021                                ;*      THAT LOADED THE PSW.
```

```
3022                                    ;*
3023                                    ;;**********************************************************************
3024  027050  000004                    TST35:  SCOPE
3025  027052  012737  027066  001110    1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3026  027060  012737  027102  000250            MOV     #3$,MMVEC       ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
3027  027066  012737  040000  177776  2$:     MOV     #40000,PSW      ;SET 01 IN PSW CURRENT MODE BITS
3028  027074  000240                            NOP                     ;FETCH OF THIS INSTRUCTION SHOULD CAUSE ABORT
3029  027076  104056                            ERROR   56              ;ILLEGAL MODE 01 NOT ABORTED
3030                                                                    ;FOR A TIGHTER SCOPE LOOP
3031                                                                    ;REPLACE ERROR CALL WITH
3032                                                                    ;"BR 2$" = 000773
3033  027100  000424                            BR      5$              ;BRANCH AROUND SR0 & SR2 CHECKS
3034  027102  012706  001100            3$:     MOV     #KERSTK,SP      ;RESTORE STACK POINTER
3035  027106  012701  100043                    MOV     #100043,R1      ;LOAD EXPECTED CONTENTS OR SR0 INTO R1
3036  027112  013737  177572  001362            MOV     SR0,WASSR0      ;READ CONTENTS OF SR0
3037
3038  027120  020137  001362                    CMP     R1,WASSR0       ;DID SR0 REPORT ILLEGAL MODE CORRECTLY?
3039  027124  001401                            BEQ     4$              ;BRANCH IF YES
3040  027126  104057                            ERROR   57              ;SR0 DID NOT REPORT NR ABORT, PG=1, MODE=01
3041                                                                    ;FOR TIGHTER SCOPE LOOP
3042                                                                    ;REPLACE ERROR CALL WITH
3043                                                                    ;"BR 2$" = 000757
3044  027130  012701  027066            4$:     MOV     #2$,R1          ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
3045  027134  013737  177576  001364            MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
3046  027142  020137  001364                    CMP     R1,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR OF ABORTED INST.
3047  027146  001401                            BEQ     5$              ;BRANCH IF YES
3048  027150  104036                            ERROR   36              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
3049                                                                    ;FOR TIGHTER SCOPE LOOP
3050                                                                    ;REPLACE ERROR CALL WITH
3051                                                                    ;"BR 2$" = 000746
3052  027152  042737  160000  177572  5$:     BIC     #160000,SR0     ;CLEAR POSSIBLE ERROR BITS IN SR0
3053  027160  012737  002314  000250            MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. ABORT VECTOR
3054  027166  012737  027052  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067                                    ;**********************************************************************
3068                                    ;*
3069                                    ;*      THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
3070                                    ;*      COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
3071                                    ;*      AND STATUS LOGIC.  THE PAGE LENGTH FIELD (PLF) IN KERNEL
3072                                    ;*      PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
3073                                    ;*      ADDRESSES ARE READ.  WHILE USING BOTH UPWARD & DOWNWARD PAGE
3074                                    ;*      EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
3075                                    ;*      "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T.
3076                                    ;*
3077                                    ;*      STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
```

```
                                        ;*      ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
                                        ;*      THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
                                        ;*      IS LOCKED UP.
                                        ;*
                                        ;;*****************************************************************


                                        ;;*****************************************************************
                                        ;*TEST 36      PAGE LENGTH FAULTS-UPWARD EXPANSION
                                        ;*
                                        ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
                                        ;*      FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
                                        ;*      ARE ACCESSED.  WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
                                        ;*      NO ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
                                        ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
                                        ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
                                        ;*      (BIT 3) OF PDR 4 = 0).
                                        ;*
                                        ;;*****************************************************************
027174  000004                  T36:    SCOPE
027176  012737  077406  172306    1$:   MOV     #77406,KIPDR3   ;MAKE SURE PDR3 IS DESCRIBED AS R/W
027204  012737  077406  172312          MOV     #77406,KIPDR5   ;MAKE SURE PDR5 IS DESCRIBED AS R/W
027212  012700  100000              MOV     #100000,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
027216  012704  000406              MOV     #406,R4         ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
027222  012737  027414  000250    2$:   MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
027230  010437  172310              MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
027234  012737  027242  001110      MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
027242  012706  001100    3$:       MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
027246  011001                      MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
027250  062700  000100              ADD     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
027254  012737  027262  001110      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
027262  012706  001100    4$:       MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
027266  011001                      MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
027270  062700  000100              ADD     #100,R0         ;FORM NEXT VIRTUAL ADDR IN R0
027274  020027  117700              CMP     R0,#117700      ;HAVE ALL PLF'S BEEN TESTED YET?
027300  001470                      BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
027302  012737  027316  001110      MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
027310  012737  027324  000250      MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
027316  011001            5$:       MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
027320  104033                      ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
                                                            ;FOR TIGHTER SCOPE LOOP
                                                            ;REPLACE ERROR CALL WITH
                                                            ;"BR 5$" = 000776
027322  000424                      BR      8$              ;BRANCH AROUND ABORT CHECKS
027324  012706  001100    6$:       MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
027330  013737  177572  001362      MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
027336  013737  177576  001364      MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
027344  012702  040011              MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
027350  020237  001362              CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
027354  001401                      BEQ     7$              ;BRANCH IF YES
027356  104034                      ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
                                                            ;FOR TIGHTER SCOPE LOOP
                                                            ;REPLACE ERROR CALL WITH
                                                            ;"BR 5$" = 000757
027360  012703  027316    7$:       MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
```

```
3134  027364  020337  001364          CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3135  027370  001401                  BEQ     8$              ;BRANCH IF YES
3136  027372  104035                  ERROR   35              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3137                                                          ;FOR TIGHTER SCOPE LOOP
3138                                                          ;REPLACE ERROR CALL WITH
3139                                                          ;"BR 5$" = 000751
3140  027374  042737  160000  177572  8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3141  027402  062704  000400          ADD     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
3142  027406  162700  000100          SUB     #100,R0         ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
3143  027412  000703                  BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3144                                                          ;THE PLF VALUE JUST FORMED
3145  027414  012637  001356  9$:     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3146  027420  012637  001360          MOV     (KSP)+,TRAPPS
3147  027424  013737  177572  001362  MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
3148  027432  013737  177576  001364  MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3149  027440  042737  160000  177572  BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3150  027446  104032                  ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3151                                                          ;FOR TIGHTER SCOPE LOOP
3152                                                          ;REPLACE ERROR CALL WITH
3153                                                          ;A "NOP" = 000240
3154  027450  013746  001360          MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3155  027454  013746  001356          MOV     TRAPPC,-(KSP)
3156  027460  000002                  RTI                     ;RETURN FROM UNEXPECTED ABORT
3157
3158  027462  012737  027176  001110  10$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3159  027470  012737  002314  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3160                                                          ;ADDRESS TO M.M. TRAP VECTOR
3161
3162
3163                          ;*****************************************************************
3164                          ;*TEST 37       PAGE LENGTH FAULTS-DOWNWARD EXPANSION
3165                          ;*
3166                          ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
3167                          ;*      FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
3168                          ;*      ARE ACCESSED.  WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
3169                          ;*      NO PAGE ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS LESS THAN PDR <14:8>
3170                          ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
3171                          ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
3172                          ;*      (BIT 3) OF PDR4=1).
3173                          ;*
3174                          ;*****************************************************************
3175  027476  000004          TST37:  SCOPE
3176  027500  012700  117700  1$:     MOV     #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3177  027504  012704  077016          MOV     #77016,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
3178  027510  012737  027702  000250  2$:     MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3179  027516  010437  172310          MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3180  027522  012737  027530  001110  MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3181  027530  012706  001100  3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3182  027534  011001                  MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
3183  027536  162700  000100          SUB     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
3184  027542  012737  027550  001110  MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3185  027550  012706  001100  4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3186  027554  011001                  MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3187  027556  162700  000100          SUB     #100,R0         ;FORM NEXT VIRTUAL ADDR. IN R0
3188  027562  020027  100000          CMP     R0,#100000      ;HAVE ALL PLF'S BEEN TESTED YET?
3189  027566  001470                  BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
```

```
3190  027570  012737  027604  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3191  027576  012737  027612  000250          MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3192  027604  011001                  5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
3193  027606  104033                          ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3194                                                                  ;FOR TIGHTER SCOPE LOOP
3195                                                                  ;REPLACE ERROR CALL WITH
3196                                                                  ;"BR 5$" = 000776
3197  027610  000424                          BR      8$              ;BRANCH AROUND ABORT CHECKS
3198  027612  012706  001100          6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
3199  027616  013737  177572  001362          MOV     SR0,WAS$R0      ;READ M.M. STATUS REG. 0
3200  027624  013737  177576  001364          MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
3201  027632  012702  040011                  MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
3202  027636  020237  001362                  CMP     R2,WAS$R0       ;DID SR0 REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
3203  027642  001401                          BEQ     7$              ;BRANCH IF YES
3204  027644  104034                          ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3205                                                                  ;FOR TIGHTER SCOPE LOOP
3206                                                                  ;REPLACE ERROR CALL WITH
3207                                                                  ;"BR 5$" = 000757
3208  027646  012703  027604          7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
3209  027652  020337  001364                  CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3210  027656  001401                          BEQ     8$              ;BRANCH IF YES
3211  027660  104035                          ERROR   35              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3212                                                                  ;FOR TIGHTER SCOPE LOOP
3213                                                                  ;REPLACE ERROR CALL WITH
3214                                                                  ;"BR 5$" = 000751
3215  027662  042737  160000  177572  8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3216  027670  162704  000400                  SUB     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
3217  027674  062700  000100                  ADD     #100,R0         ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
3218  027700  000703                          BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3219                                                                  ;THE PLF VALUE JUST FORMED
3220  027702  012637  001356          9$:     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3221  027706  012637  001360                  MOV     (KSP)+,TRAPPS
3222  027712  013737  177572  001362          MOV     SR0,WA$SR0      ;SAVE CONTENTS OF SR0 FOR ERROR
3223  027720  013737  177576  001364          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3224  027726  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3225  027734  104032                          ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3226                                                                  ;FOR TIGHTER SCOPE LOOP
3227                                                                  ;REPLACE ERROR CALL WITH
3228                                                                  ;A "NOP" = 000240
3229  027736  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3230  027742  013746  001356                  MOV     TRAPPC,-(KSP)
3231  027746  000002                          RTI                     ;RETURN FROM UNEXPECTED ABORT
3232
3233  027750  012737  027500  001110  10$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3234  027756  012737  002314  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3235                                                                  ;ADDRESS TO M.M. TRAP VECTOR
3236
3237
3238                                  ;;*****************************************************************
3239                                  ;*TEST 40       SR2 BIT TEST
3240                                  ;*
3241                                  ;*      THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
3242                                  ;*      CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 100000
3243                                  ;*      TO 111000 (PHYSICAL ADDRESSES 060000-071000).  KIPDR4 IS USED TO EXECUTE
3244                                  ;*      THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
3245                                  ;*      010727  MOV     PC,(PC)+  ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
```

```
3246                                    ;*      000000                  ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
3247                                    ;*      000137  JMP     @#3$    ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
3248                                    ;*      (ADDR. OF 3$)           ;IN CASE A R/O ABORT DOES NOT OCCUR,
3249                                    ;*                              ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
3250                                    ;*
3251                                    ;;*****************************************************************
3252  027764  000004                    TST40:  SCOPE
3253  027766  012737  000600  172346    1$:     MOV     #600,KIPAR3     ;BE SURE PAR3 IS MAPPED TO 12-16K
3254  027774  012737  000600  172350            MOV     #600,KIPAR4     ;BE SURE PAR4 IS MAPPED TO 12-16K
3255  030002  012737  077406  172306            MOV     #77406,KIPDR3   ;MAP PAGE 3  128 BLOCKS, R/W
3256  030010  012737  077402  172310            MOV     #77402,KIPDR4   ;MAP PAGE 4  128 BLOCKS, READ-ONLY
3257  030016  012700  060000                    MOV     #60000,R0       ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
3258  030022  012701  100000                    MOV     #100000,R1      ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
3259  030026  012737  030062  000250            MOV     #3$,MMVEC       ;SET M.M. TRAP VECTOR TO 3$
3260  030034  012737  030042  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3261  030042  012720  010727            2$:     MOV     #010727,(R0)+   ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR.
3262  030046  005020                            CLR     (R0)+           ;       REACHED THRU PDR/PAR 4.
3263  030050  012720  000137                    MOV     #000137,(R0)+   ;LOAD "JMP @#3$" INSTRUCTION AT VIRT. ADDR.
3264  030054  012710  030062                    MOV     #3$,(R0)        ;       IN CASE R/O VIOL. DOES NOT ABORT
3265  030060  010107                            MOV     R1,PC           ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3266  030062  012706  001100            3$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER
3267  030066  013737  177576  001364            MOV     SR2,WASSR2      ;READ CONTENTS OF STATUS REG 2
3268  030074  020137  001364                    CMP     R1,WASSR2       ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
3269  030100  001401                            BEQ     4$              ;BRANCH IF YES
3270  030102  104036                            ERROR   36              ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
3271                                                                    ;FOR TIGHTER SCOPE LOOP
3272                                                                    ;REPLACE ERROR CALL WITH
3273                                                                    ;"BR 2$" = 000757
3274  030104  042737  160000  177572  4$:     BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
3275  030112  162700  000004                    SUB     #4,R0           ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
3276  030116  062701  000002                    ADD     #2,R1           ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
3277  030122  020127  111002                    CMP     R1,#111002      ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
3278  030126  103745                            BLO     2$              ;BRANCH IF NO
3279
3280  030130  012737  027766  001110  5$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3281  030136  012737  077406  172310            MOV     #77406,KIPDR4   ;RESTORE PDR4 TO R/W ACCESS
3282  030144  012737  002314  000250            MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL M.M.
3283                                                                    ;TRAP HANDLER TO M.M. VECTOR
3284
3285
3286                                    ;;*****************************************************************
3287                                    ;*TEST 41        MORE CHECKS OF SR0 & SR2
3288                                    ;*
3289                                    ;*      THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SR0 & SR2 LOGIC.
3290                                    ;*      FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS
3291                                    ;*      PROGRAM COUNTER.  ALSO SR0 & SR2 ARE LOCKED UP BY A PAGE LENGTH
3292                                    ;*      ABORT, THEN WITHOUT CLEARING SR0'S ERROR BITS, A R/O ABORT IS CAUSED.
3293                                    ;*      SR0 & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
3294                                    ;*      INFORMATION ABOUT THE PAGE LENGTH ABORT SOULD STILL BE LOCKED UP.
3295                                    ;*      IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SR0 IS CLEARED
3296                                    ;*      AND SR2 IS UNLOCKED BY A RESET.  AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
3297                                    ;*      SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
3298                                    ;*
3299                                    ;;*****************************************************************
3300  030152  000004                    TST41:  SCOPE
3301  030154  012737  000600  172352    1$:     MOV     #600,KIPAR5     ;MAP KERNEL PAGE 5 TO 12-16K
```

```
3302  030162  012737  000406  172310          MOV     #406,KIPDR4     ;SETUP PDR4 FOR PAGE LENGTH ABORT
3303  030170  012737  077402  172312          MOV     #77402,KIPDR5   ;SETUP PDR5 FOR R/O ABORT
3304  030176  012737  030204  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3305  030204  013737  177576  001364  2$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
3306  030212  012701  030204                  MOV     #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
3307  030216  020137  001364                  CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
3308  030222  001401                          BEQ     3$              ;BRANCH IF YES
3309  030224  104041                          ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3310                                                                  ;FOR TIGHTER SCOPE LOOP
3311                                                                  ;REPLACE ERROR CALL WITH
3312                                                                  ;"BR 2$" = 000767
3313  030226  012737  030234  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3314  030234  013737  177576  001364  4$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
3315  030242  012701  030234                  MOV     #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
3316  030246  020137  001364                  CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 4$
3317  030252  001401                          BEQ     5$              ;BRANCH IF YES
3318  030254  104041                          ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3319                                                                  ;FOR TIGHTER SCOPE LOOP
3320                                                                  ;REPLACE ERROR CALL WITH
3321                                                                  ;"BR 4$" = 000767
3322  030256  012737  030264  001110  5$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
3323  030264  012737  030302  000250  6$:     MOV     #7$,MMVEC       ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
3324  030272  005037  001200                  CLR     $TMP1           ;CLEAR ERROR INDICATOR
3325  030276  005237  100500                  INC     @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
3326  030302  012706  001100          7$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
3327  030306  013737  177572  001176          MOV     SR0,$TMP0       ;SAVE SR0'S INFORMATION ON PG. LGTH. ABORT
3328  030314  013737  177576  001202          MOV     SR2,$TMP2       ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3329  030322  012737  030334  000250          MOV     #8$,MMVEC       ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
3330  030330  005237  120000                  INC     @#120000        ;CAUSE R/O ABORT - TRAP TO 8$
3331  030334  012706  001100          8$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
3332  030340  013737  177572  001362          MOV     SR0,WASSR0      ;READ SR0 FOLLWOING SECOND KT ABORT
3333  030346  013737  177576  001364          MOV     SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
3334  030354  023737  001176  001362          CMP     $TMP0,WASSR0    ;IS SR0 STILL HOLDING INFO ON FIRST ABORT?
3335  030362  001402                          BEQ     9$              ;BRANCH IF YES
3336  030364  005237  001200                  INC     $TMP1           ;SET ERROR INDICATOR
3337  030370  023737  001202  001364  9$:     CMP     $TMP2,WASSR2    ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
3338  030376  001402                          BEQ     10$             ;BRANCH IF YES
3339  030400  005237  001200                  INC     $TMP1           ;SET ERROR INDICATOR
3340  030404  005737  001200          10$:    TST     $TMP1           ;WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
3341  030410  001401                          BEQ     11$             ;BRANCH IF NO
3342  030412  104037                          ERROR   37              ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3343                                                                  ;FOR TIGHTER SCOPE LOOP
3344                                                                  ;REPLACE ERROR CALL WITH
3345                                                                  ;"BR 6$" = 000726
3346  030414  005037  001200          11$:    CLR     $TMP1           ;CLEAR ERROR INDICATOR
3347  030420  000005                          RESET                   ;EXECUTE A RESET, APPLYING AN "INIT"
3348  030422  013737  177572  001362          MOV     SR0,WASSR0      ;READ SR0
3349  030430  005737  001362                  TST     WASSR0          ;WAS SR0 CLEARED BY THE RESET?
3350  030434  001402                          BEQ     12$             ;BRANCH IF YES
3351  030436  005237  001200                  INC     $TMP1           ;SR0 NOT CLEARED BY A RESET
3352  030442  013737  177576  001364  12$:    MOV     SR2,WASSR2      ;READ SR2
3353  030450  022737  030442  001364          CMP     #12$,WASSR2     ;WAS SR2 UNLOCKED BY A RESET?
3354  030456  001402                          BEQ     13$             ;BRANCH IF YES
3355  030460  005237  001200                  INC     $TMP1           ;SR2 NOT UNLOCKED BY A RESET
3356  030464  005737  001200          13$:    TST     $TMP1           ;WERE SR0 & SR2 BOTH "RESET" BY A RESET?
3357  030470  001401                          BEQ     14$             ;BRANCH IF YES
```

```
3358  030472  104040                          ERROR   40          ;SR0 OR SR2 NOT "RESET" BY A RESET
3359                                                              ;FOR TIGHTER SCOPE LOOP
3360                                                              ;REPLACE ERROR CALL WITH
3361                                                              ;"BR 6$" = 000676
3362  030474  005237  177572          14$:    INC     SR0         ;TURN MEMORY MANAGEMENT BACK ON
3363  030500  013737  177576  001364  15$:    MOV     SR2,WASSR2  ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3364  030506  012701  030500                  MOV     #15$,R1     ;PUT EXPECTED VIRTUAL PC IN R1
3365  030512  020137  001364                  CMP     R1,WASSR2   ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3366  030516  001401                          BEQ     16$         ;BRANCH IF YES
3367  030520  104041                          ERROR   41          ;SR2 NOT TRACKING CORRECTLY
3368                                                              ;FOR TIGHTER SCOPE LOOP
3369                                                              ;REPLACE ERROR CALL WITH
3370                                                              ;"BR 6$" = 000663
3371  030522  012737  030154  001110  16$:    MOV     #1$,$LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
3372  030530  012737  077406  172310          MOV     #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
3373  030536  012737  077406  172312          MOV     #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
3374  030544  012737  002314  000250          MOV     #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
3375                                                              ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
3376
3377
3378                          ;*****************************************************************
3379                          ;*TEST 42       USER ABORT PICKS UP KERNEL SPACE VECTOR
3380                          ;*
3381                          ;*      THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
3382                          ;*      USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
3383                          ;*      KERNEL SPACE.  USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
3384                          ;*      THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
3385                          ;*      WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
3386                          ;*      SHOULD BE PICKED UP FROM LOC. 000004.  AN ODD ADDRESS ERROR IS USED
3387                          ;*      TO CAUSE A TRAP TO "4".
3388                          ;*
3389                          ;*****************************************************************
3390  030552  000004          TST42:  SCOPE
3391  030554  004737  035432  1$:     JSR     PC,TOFF     ;TURN OFF T-BIT TRAPPING FOR THIS TEST
3392  030560  012737  030566  001110          MOV     #2$,$LPERR  ;SET LOOP ON ERROR POINTER TO 2$
3393  030566  005037  177776          2$:     CLR     PSW         ;GO TO KERNEL MODE
3394  030572  012706  001100                  MOV     #KERSTK,KSP ;SETUP KERNEL STACK PTR.
3395  030576  012737  000600  177640          MOV     #600,UIPAR0 ;MAP USER PAGE 0 TO 12K
3396  030604  012737  030666  000004          MOV     #4$,@#4     ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
3397  030612  012737  000340  000006          MOV     #340,@#6    ;LOAD VECTOR+2 WITH NEW PSW
3398  030620  012737  140000  177776          MOV     #140000,PSW ;GO TO USER MODE
3399  030626  012706  000700                  MOV     #USESTK,USP ;SETUP USER STACK PTR.
3400  030632  012737  030652  000004          MOV     #3$,@#4     ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
3401  030640  012737  000340  000006          MOV     #340,@#6    ;LOAD VECTOR+2 WITH NEW PSW
3402  030646  005737  030653                  TST     3$+1        ;CAUSE ODD ADDR. ERROR TRAP TO "4"
3403                                                              ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
3404                                                              ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
3405  030652  013701  177776          3$:     MOV     PSW,R1      ;SAVE PSW FOR ERROR
3406  030656  010602                          MOV     SP,R2       ;SAVE VALUE OF STACK POINTER FOR ERROR
3407  030660  005037  177776                  CLR     PSW         ;BE SURE BACK IN KERNEL MODE
3408  030664  104042                          ERROR   42          ;DID NOT TRAP THRU KERNEL SPACE
3409                                                              ;FOR TIGHTER SCOPE LOOP
```

```
3410                                                    ;REPLACE ERROR CALL WITH
3411                                                    ;"BR 2$" = 000740
3412  030666  005037  177776          4$:     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
3413  030672  012706  001100                  MOV     #KERSTK,KSP     ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3414  030676  005037  177640                  CLR     UIPAR0          ;REMAP USER PAGE 0 TO 0-4K
3415  030702  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
3416  030710  012706  000700                  MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
3417  030714  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
3418  030720  012737  002246  000004          MOV     #TIMERR,@#4     ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3419  030726  012737  030554  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3420  030734  004737  035466                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
3421
3422                                          ;;*****************************************************************
3423                                          ;*TEST 43       RTI IN USER MODE DOES NOT CHANGE PSW
3424                                          ;*
3425                                          ;*      THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3426                                          ;*      MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3427                                          ;*
3428                                          ;;*****************************************************************
3429  030740  000004                  TST43:  SCOPE
3430
3431  030742  012737  030754  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3432  030750  012702  170000                  MOV     #170000,R2      ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
3433  030754  010237  177776          2$:     MOV     R2,PSW          ;GO TO USER MODE-PRIORITY 0
3434  030760  012746  000340                  MOV     #340,-(SP)      ;PUT A NEW PSW (PRIORITY=7) ON STACK
3435  030764  012746  030772                  MOV     #3$,-(SP)       ;PUT NEW PC ON THE STACK
3436  030770  000002                          RTI                     ;DO AN RTI FROM USER MODE
3437  030772  013701  177776          3$:     MOV     PSW,R1          ;READ NEW PSW INTO R1
3438  030776  042701  007437                  BIC     #7437,R1        ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3439  031002  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
3440  031006  020201                          CMP     R2,R1           ;DID PSW STAY IN USER, PRIORITY=0?
3441  031010  001401                          BEQ     4$              ;BRANCH IF YES
3442  031012  104060                          ERROR   60              ;PSW CHANGED BY AN RTI FROM USER
3443                                                                  ;FOR A TIGHTER SCOPE LOOP
3444                                                                  ;REPLACE ERROR CALL WITH
3445                                                                  ;"BR=2$" = 000760
3446  031014  012737  030742  001110  4$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3447
3448
```

```
                        ;;*****************************************************************
                        ;*TEST 44       KT ERROR NOT SERVICED IF ODD ADDR. ERROR
                        ;*
                        ;*      THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
                        ;*      WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS
                        ;*      ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY
                        ;*      MANAGEMENT ERROR ISN'T.  THIS MEANS THAT SR0 AND SR2
                        ;*      SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
                        ;*      A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
                        ;*      ERROR
                        ;*
                        ;;*****************************************************************
031022  000004          TST44:  SCOPE
031024  012737  000600  172350  1$:     MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
031032  012705  077402          MOV     #77402,R5       ;LOAD PDR4 DATA INTO R5
031036  010537  172310          MOV     R5,KIPDR4       ;MAP PAGE 4 READ-ONLY
031042  012737  031072  000004  MOV     #4$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
031050  012737  031070  000250  MOV     #3$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
031056  012737  031064  001110  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
031064  005237  060001          2$:     INC     60001           ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
031070  104043                  3$:     ERROR   43              ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000776
031072  012706  001100          4$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER TRAPPING
031076  005037  001200          CLR     $TMP1           ;CLEAR ERROR INDICATOR
031102  013737  177572  001362  MOV     SR0,WASSR0      ;READ STATUS REG. 0
031110  013737  177576  001364  5$:     MOV     SR2,WASSR2      ;READ STATUS REG. 2
031116  012700  000017          MOV     #17,R0          ;LOAD EXPECTED SR0 CONTENTS INTO R0
031122  020037  001362          CMP     R0,WASSR0       ;SR0  ERROR BITS LEFT CLEAR BY TRAPPING?
031126  001402                  BEQ     6$              ;BRANCH IF YES
031130  005237  001200          INC     $TMP1           ;SR0 ERROR BITS SET WHEN ODD ADDR. SERVICED
031134  012701  031110          6$:     MOV     #5$,R1          ;LOAD EXPECTED SR2 CONTENTS INTO R1
031140  020137  001364          CMP     R1,WASSR2       ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
031144  001402                  BEQ     7$              ;BRANCH IF YES
031146  005237  001200          INC     $TMP1           ;SR2 LOCKED UP BY ODD ADDR. ERROR
031152  005737  001200          7$:     TST     $TMP1           ;WHERE SR0 OR SR2 EFFECTED?
031156  001404                  BEQ     8$              ;BRANCH IF NO
031160  104044                  ERROR   44              ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000741
031162  042737  160000  177572  BIC     #160000,SR0     ;CLEAR ERROR BITS THAT MAY BE SET IN SR0
031170  012737  002246  000004  8$:     MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
031176  012737  002314  000250  MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
031204  012737  077406  172310  MOV     #77406,KIPDR4   ;REMAP PAGE 4 TO READ/WRITE
031212  012737  031024  001110  MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$


                        ;;*****************************************************************
                        ;*TEST 45       PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR. ERROR
                        ;*
                        ;*      THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
                        ;*      A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
                        ;*      SERVICING OF AN ODD ADDR. ERROR.  DURING A "DOUBLE ERROR"
                        ;*      SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
```

```
                                        ;*      FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP.
                                        ;*      NOT THE PSW PRESENT BEFORE THE FIRST TRAP.  SR0 AND SR2
                                        ;*      SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
                                        ;*
                                        ;*      NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
                                        ;*      WILL BE SET IN THE PSW THAT IS SAVED.
                                        ;*
                                        ;**********************************************************************
031220  000004                    TST45:  SCOPE
031222  004737  035432            1$:     JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
031226  012737  000600  177646            MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12-16K
031234  012737  000600  177650            MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12-16K
031242  012737  077402  177606            MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
031250  012737  077406  177610            MOV     #77406,UIPDR4   ;MAP USER PAGE 4 READ/WRITE
031256  012737  031332  000004            MOV     #4$,@#4         ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
031264  012737  140017  000006            MOV     #140017,@#6     ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
031272  012737  031332  000250            MOV     #4$,@#250       ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
031300  012737  000340  000252            MOV     #340,@#252      ;LOAD A KERNEL PSW IN MMVEC+2
031306  012737  031314  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
031314  012737  140000  177776  2$:       MOV     #140000,PSW     ;GO TO USER MODE
031322  012706  100002                    MOV     #100002,USP     ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
031326  005237  100005          3$:       INC     100005          ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
                                                                  ;R/O ERROR WHEN TRY TO SAVE OLD PC
031332  016601  000002          4$:       MOV     2(KSP),R1       ;PUT PSW SAVED ON KERNEL STACK INTO R1
031336  011603                            MOV     (KSP),R3        ;PUT PC SAVED ON KERNEL STACK INTO R3
031340  013737  177572  001362            MOV     SR0,WASSR0      ;READ THE CONTENTS OF M.M. STATUS REG. 0
031346  013737  177576  001364            MOV     SR2,WASSR2      ;READ THE CONTENTS OF M.M. STATUS REG. 2
031354  042737  160000  177572            BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
031362  005037  177776                    CLR     PSW             ;BE SURE IN KERNEL MODE
031366  012706  001100                    MOV     #KERSTK,KSP     ;RESTORE KERNEL STACK POINTER
031372  012737  140000  177776            MOV     #140000,PSW     ;GO TO USER MODE
031400  012706  000700                    MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
031404  005037  177776                    CLR     PSW             ;GO BACK TO KERNEL MODE
031410  005037  001176                    CLR     $TMP0           ;CLEAR ERROR INDICATOR
031414  020127  170017                    CMP     R1,#170017      ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
                                                                  ;ODD ADDR. TRAP FROM ERRVEC+2?
                                                                  ;VALUE 170017 = PSW FROM LOC. 6 WITH
                                                                  ;PREVIOUS MODE BITS = USER
031420  001402                    BEQ     5$              ;BRANCH IF YES
031422  005237  001176            INC     $TMP0           ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
```

```
3545  031426  020327  031332          5$:     CMP     R3,#3$+4        ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
3546                                                                    ;SAVED ON THE STACK?
3547  031432  001402                          BEQ     6$              ;BRANCH IF YES
3548  031434  005237  001176                  INC     $TMP0           ;WRONG PC SAVED DURING TRAP SEQUENCE
3549  031440  023727  001362  020147  6$:     CMP     WASSR0,#20147   ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
3550  031446  001402                          BEQ     7$              ;BRANCH IF YES
3551  031450  005237  001176                  INC     $TMP0           ;SR0 DID NOT REPORT R/O ABORT
3552  031454  023727  001364  031326  7$:     CMP     WASSR2,#3$      ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3553                                                                    ;INSTRUCTION SUCCESSFULLY FETCHED?
3554  031462  001402                          BEQ     8$              ;BRANCH IF YES
3555  031464  005237  001176                  INC     $TMP0           ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
3556  031470  005737  001176          8$:     TST     $TMP0           ;ANY "ERRORS" DURING TRAP SEQUENCE?
3557  031474  001401                          BEQ     9$              ;BRANCH IF NO
3558  031476  104045                          ERROR   45              ;THE WRONG PC OR PSW WERE SAVED
3559                                                                    ;OR SR0 OR SR2 DID NOT REPORT R/O
3560                                                                    ;ERROR DURING ODD ADDR. - KT TRAP
3561                                                                    ;SEQUENCE
3562                                                                    ;FOR TIGHTER SCOPE LOOP
3563                                                                    ;REPLACE ERROR CALL WITH
3564                                                                    ;"BR 2$" = 000710
3565  031500  012737  002246  000004  9$:     MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3566  031506  012737  000340  000006          MOV     #340,@#6        ;RELOAD ERRVEC+2 WITH KERNEL PSW
3567  031514  012737  002314  000250          MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3568  031522  012737  077406  177606          MOV     #77406,@IPDR3   ;REMAP USER PAGE 3 READ/WRITE
3569  031530  012737  031222  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3570  031536  004737  035466                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
3571                                  ;;**********************************************************************
3572                                  ;*
3573                                  ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3574                                  ;*      THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS.
3575                                  ;*
3576                                  ;;**********************************************************************
3577
3578
3579                                  ;;**********************************************************************
3580                                  ;*TEST 46       MOVE FROM PREVIOUS (USER) I-SPACE
3581                                  ;*
3582                                  ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3583                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3584                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3585                                  ;*
3586                                  ;*
3587                                  ;*      IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3588                                  ;*      WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
3589                                  ;*
3590                                  ;;**********************************************************************
3591  031542  000004                  T$T46:  SCOPE
3592  031544  005037  172340          1$:     CLR     KIPAR0          ;MAP KERNEL PAGE 0 TO 0-4K
3593  031550  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL PAGE 1 TO 4-8K
3594  031556  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL PAGE 2 TO 8-12K
3595  031564  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL PAGE 3 TO 12-16K
3596  031572  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
3597  031600  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL PAGE 7 TO THE I/O PAGE
3598  031606  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3599                                                                    ;READ/WRITE, LENGTH 200 BLOCKS
3600  031612  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
```

```
3601  031616  012701  172300          MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3602  031622  010021                  2$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3603  031624  077202                          SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL PDRS LOADED
3604  031626  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
3605  031632  012701  177600                  MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3606  031636  010021                  3$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3607  031640  077202                          SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL PDRS LOADED
3608  031642  012737  000000  177640          MOV     #000,UIPAR0     ;MAP USER I PAGE 0 TO 0-4K
3609  031650  012737  000200  177642          MOV     #200,UIPAR1     ;MAP USER I PAGE 1 TO 4-8K
3610  031656  012737  000400  177644          MOV     #400,UIPAR2     ;MAP USER I PAGE 2 TO 8-12K
3611  031664  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER I PAGE 3 TO 12-16K
3612  031672  012737  007600  177656          MOV     #7600,UIPAR7    ;MAP USER I PAGE 7 TO THE I/O PAGE
3613  031700  012737  031706  001110          MOV     #4$,$LPERR      ;SET LOOP ON ERROR TO 4$
3614  031706                  4$:
3615  031706  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3616  031714  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3617  031722  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3618  031730  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3619  031734  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3620  031740  012737  032342  000250          MOV     #23$,MMVEC      ;SET M.M. VECTOR TO 23$
3621  031746  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3622                                          ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3623                                          ;
3624  031752  012737  031760  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3625  031760  012737  030340  177776  5$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3626  031766  006506                  6$:     MFPI    USP             ;PUT USER STACK POINTER ON KERNEL
3627                                                                  ;STACK
3628  031770  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
```

```
3629  031774  001407                BEQ     7$              ;BRANCH IF NOTHING WAS PUSHED
3630  031776  012600                MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3631  032000  012701  000700        MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
```

```
3632  032004  020001                        CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3633  032006  001403                        BEQ     8$              ;BRANCH IF YOU DID
3634  032010  104046                        ERROR   46              ;WRONG THING WAS PUSHED ON STACK
3635                                                                ;FOR TIGHTER SCOPE LOOP
3636                                                                ;REPLACE ERROR CALL WITH
3637                                                                ;"BR 5$" = 000763
3638  032012  000401                        BR      8$              ;BRANCH TO NEXT TRY
3639  032014  104050              7$:       ERROR   50              ;NOTHING PUSHED ON STACK
3640                                                                ;FOR TIGHTER SCOPE LOOP
3641                                                                ;REPLACE ERROR CALL WITH
3642                                                                ;"BR 5$" = 000761
3643  032016              8$:       ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3644  032016  012737  032030  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3645  032024  012700  036514                MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3646  032030  012737  030340  177776  9$:   MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3647  032036  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3648  032042  006512                        MFPI    (R2)            ;READ FROM PHYSICAL 60000
3649  032044  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3650  032046  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3651  032050  001401                        BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3652  032052  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3653                                                                ;FOR TIGHTER SCOPE LOOP
3654                                                                ;REPLACE ERROR CALL WITH
3655                                                                ;"BR 9$" = 000766
3656  032054              10$:      ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3657  032054  012737  032062  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3658  032062  012737  030340  177776  11$:   MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3659  032070  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3660  032074  006522                        MFPI    (R2)+           ;READ FROM PHYSICAL 60000
3661  032076  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3662  032100  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3663  032102  001401                        BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3664  032104  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3665                                                                ;FOR TIGHTER SCOPE LOOP
3666                                                                ;REPLACE ERROR CALL WITH
3667                                                                ;"BR 11$" = 000766
3668  032106              12$:      ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
```

```
 3669  032106  012737  032114  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
 3670  032114  012737  030340  177776  13$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
 3671  032122  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
 3672  032126  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 3673  032130  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
 3674  032132  001401                          BEQ     14$             ;BRANCH IF CORRECT DATA WAS FETCHED
 3675  032134  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
 3676                                                                  ;FOR TIGHTER SCOPE LOOP
 3677                                                                  ;REPLACE ERROR CALL WITH
```

```
                                                                ;"BR 13$" = 000767
032136                                  14$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
032136  012737  032144  001110                  MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
032144  012737  030340  177776  15$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
032152  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
032156  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 60000
032160  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
032162  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
032164  001401                          BEQ     16$             ;BRANCH IF CORRECT DATA WAS FETCHED
032166  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 15$" = 000766
032170                                  16$:
                                        ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
                                        ;
032170  012737  032176  001110                  MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
032176  012737  030340  177776  17$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
032204  012737  100000  001202                  MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
032212  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
032216  006552                          MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
032220  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
032222  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
032224  001401                          BEQ     18$             ;BRANCH IF CORRECT DATA WAS FETCHED
032226  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 17$" = 000763
032230                                  18$:    ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
                                        ;
032230  012737  032236  001110                  MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
032236  012737  030340  177776  19$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
032244  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
032246  006562  100000                  MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
032252  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
032254  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
032256  001401                          BEQ     20$             ;BRANCH IF CORRECT DATA WAS FETCHED
032260  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 19$" = 000766
032262                                  20$:    ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
                                        ;
032262  012737  032270  001110                  MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
032270  012737  030340  177776  21$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
032276  012737  100000  001202                  MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
032304  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
032310  006572  000000                  MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
                                                                ;ADDRESS OF 60000
032314  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
032316  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
032320  001401                          BEQ     22$             ;BRANCH IF CORRECT DATA WAS FETCHED
032322  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 21$" = 000762
```

```
3734  032324  012737  002314  000250  22$:    MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3735  032332  012737  031544  001110          MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3736  032340  000423                          BR      TST47           ;;BRANCH TO NEXT TEST
3737
3738
3739  032342  012637  001356          23$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3740  032346  012637  001360                  MOV     (KSP)+,TRAPPS
3741  032352  013737  177572  001362          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3742  032360  013737  177576  001364          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3743  032366  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
3744  032374  104051                          ERROR   51              ;TRIED TO READ NON-RESIDENT PAGE
3745                                                                  ;FOR TIGHTER SCOPE LOOP
3746                                                                  ;REPLACE ERROR CALL WITH
3747                                                                  ;A "NOP" = 000240
3748  032376  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3749  032402  013746  001356                  MOV     TRAPPC,-(KSP)
3750  032406  000002                          RTI
3751
3752
3753                                  ;;**************************************************************
3754                                  ;*TEST 47       MOVE TO PREVIOUS (USER) I-SPACE
3755                                  ;*
3756                                  ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3757                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3758                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3759                                  ;*
3760                                  ;*
3761                                  ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3762                                  ;*      WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3763                                  ;*
3764                                  ;**************************************************************
3765  032410  000004          TST47:  SCOPE
3766  032412  012737  077406  172310  1$:     MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3767  032420  012737  077406  177610          MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
3768  032426  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3769  032434  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3770  032442  012737  033222  000250          MOV     #20$,MMVEC      ;SET M.M. VECTOR TO 20$
3771                                          ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3772                                          ;
3773  032450  012737  030340  177776  2$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3774  032456  012746  007777                  MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3775  032462  006606                          MTPI    USP             ;LOAD USER STACK POINTER
3776  032464  006506                          MFPI    USP             ;READ USER STACK POINTER
3777  032466  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3778  032470  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
3779  032474  001401                          BEQ     3$              ;BRANCH IF IT WAS
3780  032476  104050                          ERROR   50              ;USER STACK POINTER NOT CHANGED
3781                                                                  ;FOR TIGHTER SCOPE LOOP
3782                                                                  ;REPLACE ERROR CALL WITH
3783                                                                  ;"BR 2$" = 000764
3784  032500  012737  030340  177776  3$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3785  032506  012746  000700                  MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER S. POINT
3786  032512  006606                          MTPI    USP             ;RESTORE USER STACK POINTER
3787  032514                          4$:     ;THIS WILL TEST DSTM = 1 MTPI.
3788  032514  012737  032532  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3789  032522  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
```

```
3790  032526  012700  125252          MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3791  032532  010046                  5$:     MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3792  032534  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3793  032540  006612                          MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3794  032542  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3795  032550  011201                          MOV     (R2),R1         ;READ FROM ADDRESS 60000
3796  032552  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
3797  032554  001401                          BEQ     6$              ;BRANCH IF STORE WAS CORRECT
3798  032556  104047                          ERROR   47              ;INCORRECT STORE
3799                                                                  ;FOR TIGHTER SCOPE LOOP
3800                                                                  ;REPLACE ERROR CALL WITH
3801                                                                  ;"BR 5$" = 000765
3802  032560                          6$:     ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3803                                          ;
3804  032560  012737  032604  001110          MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
3805  032566  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3806  032574  012700  125252                  MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3807  032600  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3808  032604  010046                  8$:     MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3809  032606  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3810  032612  006612                          MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3811  032614  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESISENT
3812  032622  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3813  032626  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3814  032630  001401                          BEQ     9$              ;BRANCH IF STORE WAS CORRECT
3815  032632  104047                          ERROR   47              ;INCORRECT STORE
3816                                                                  ;FOR TIGHTER SCOPE LOOP
3817                                                                  ;REPLACE ERROR CALL WITH
3818                                                                  ;"BR 8$" = 000764
3819  032634                          9$:     ;THIS WILL TEST DSTM = 3 MTPI.
3820  032634  012737  032654  001110          MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3821  032642  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3822  032650  012700  052525                  MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3823  032654  010046                  10$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3824  032656  105037  172310                  CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3825  032662  006637  100000                  MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
3826  032666  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3827  032674  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3828  032700  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
```

```
3829  032702  001401                        BEQ     11$             ;BRANCH IF STORE WAS CORRECT
3830  032704  104047                        ERROR   47              ;INCORRECT STORE
3831                                                                ;FOR TIGHTER SCOPE LOOP
3832                                                                ;REPLACE ERROR CALL WITH
3833                                                                ;"BR 10$" = 000763
3834  032706                          11$:  ;THIS WILL TEST DSTM = 4' MTPI.
3835  032706  012737  032726  001110        MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3836  032714  012737  030340  177776        MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3837  032722  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3838  032726  010046                  12$:  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3839  032730  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
```

```
3840  032734  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3841  032740  006642                  MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
3842  032742  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3843  032750  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3844  032754  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3845  032756  001401                  BEQ     13$             ;BRANCH IF STORE WAS CORRECT
3846  032760  104047                  ERROR   47              ;INCORRECT STORE
3847                                                          ;FOR TIGHTER SCOPE LOOP
3848                                                          ;REPLACE ERROR CALL WITH
3849                                                          ;"BR 12$" = 000762
3850  032762                  13$:    ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3851                                  ;
3852  032762  012737  033014  001110  MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
3853  032770  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3854  032776  012700  052525          MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3855  033002  012702  001204          MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3856  033006  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3857  033014  010046          14$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3858  033016  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3859  033022  006652                  MTPI    @-(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
3860  033024  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3861  033032  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3862  033036  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3863  033040  001401                  BEQ     15$             ;BRANCH IF STORE WAS CORRECT
3864  033042  104047                  ERROR   47              ;INCORRECT STORE
3865                                                          ;FOR TIGHTER SCOPE LOOP
3866                                                          ;REPLACE ERROR CALL WITH
3867                                                          ;"BR 14$" = 000764
3868  033044                  15$:    ;THIS WILL TEST DSTM = 6 MTPI.
3869                                  ;
3870  033044  012737  033066  001110  MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
3871  033052  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3872  033060  012700  052525          MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3873  033064  005002                  CLR     R2              ;MAKE REGISTER 2 ZERO
3874  033066  010046          16$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3875  033070  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3876  033074  006662  100000          MTPI    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
3877  033100  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3878  033106  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3879  033112  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3880  033114  001401                  BEQ     17$             ;BRANCH IF STORE WAS CORRECT
3881  033116  104047                  ERROR   47              ;INCORRECT STORE
3882                                                          ;FOR TIGHTER SCOPE LOOP
3883                                                          ;REPLACE ERROR CALL WITH
3884                                                          ;"BR 16$" = 000763
3885  033120                  17$:    ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3886                                  ;
3887  033120  012737  033152  001110  MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
3888  033126  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3889  033134  012700  125252          MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3890  033140  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
3891                                                          ;INTO LOCATION $TMP2
3892  033146  012702  001202          MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3893  033152  010046          18$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3894  033154  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3895  033160  006672  000000          MTPI    @0(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
```

```
3896  033164  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3897  033172  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3898  033176  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3899  033200  001401                          BEQ     19$             ;BRANCH IF STORE WAS CORRECT
3900  033202  104047                          ERROR   47              ;INCORRECT STORE
3901                                                                  ;FOR TIGHTER SCOPE LOOP
3902                                                                  ;REPLACE ERROR CALL WITH
3903                                                                  ;"BR 18$" = 000763
3904  033204  012737  032412  001110  19$:    MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3905  033212  012737  002314  000250          MOV     #MGMERR,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3906  033220  000423                          BR      TST50           ;;BRANCH TO NEXT TEST
3907
3908
3909  033222  012637  001356          20$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3910  033226  012637  001360                  MOV     (KSP)+,TRAPPS
3911  033232  013737  177572  001362          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3912  033240  013737  177576  001364          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3913  033246  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3914  033254  104051                          ERROR   51              ;TRIED TO LOAD A N.R. PAGE 4
3915                                                                  ;FOR TIGHTER SCOPE LOOP
3916                                                                  ;REPLACE ERROR CALL WITH
3917                                                                  ;A "NOP" = 000240
3918  033256  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3919  033262  013746  001356                  MOV     TRAPPC,-(KSP)
3920  033266  000002                          RTI                     ;RETURN TO TEST
3921
3922
3923
3924                                  ;;**********************************************************************
3925                                  ;*TEST 50       MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
3926                                  ;*
3927                                  ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
3928                                  ;*      FETCH IS FROM KERNEL SPACE.
3929                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3930                                  ;*
3931                                  ;*
3932                                  ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3933                                  ;*      WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
3934                                  ;*
3935                                  ;;**********************************************************************
3936  033270  000004                  TST50:  SCOPE
3937  033272  012700  077406          1$:     MOV     #77406,R0       ;MAKE ALL USER I-SPACE PAGES RESIDENT
3938                                                                  ;READ/WRITE, LENGTH 200 BLOCKS
3939  033276  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
3940  033302  012701  177600                  MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR IN R1
3941  033306  010021                  2$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3942  033310  077202                          SOB     R2,2$           ;LOOP UNTIL 8 USER PDRS LOADED
3943  033312  012737  033320  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR TO 3$
3944  033320  012737  140340  177776  3$:     MOV     #140340,PSW     ;GO TO USER MODE FOR THIS TEST
3945  033326  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3946  033334  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3947  033342  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3948  033350  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3949  033354  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3950  033360  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3951                                          ;THE FOLLOWING WILL TEST DSTM=0 MFPI
```

```
                                                ;
033364  012737  033766  000250        MOV     #21$,MMVEC      ;SET M.M. VECTOR TO 21$
033372  105037  177610                CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
033376  012737  140340  177776        MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
033404  006506                4$:     MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
033406  022706  000700                CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT 1$
033412  001407                        BEQ     5$              ;BRANCH IF NOTHING WAS PUSHED
033414  012600                        MOV     (USP)+,R0       ;POP USER STACK INTO R0
033416  012701  001100                MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
033422  020001                        CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
033424  001403                        BEQ     6$              ;BRANCH IF YOU DID
033426  104046                        ERROR   46              ;WRONG THING WAS PUSHED ON STACK
                                                ;FOR TIGHTER SCOPE LOOP
                                                ;REPLACE ERROR CALL WITH
                                                ;"BR 4$" = 000766
033430  000401                        BR      6$              ;BRANCH TO NEXT TRY
033432  104050                5$:     ERROR   50              ;NOTHING PUSHED ON STACK
                                                ;FOR TIGHTER SCOPE LOOP
                                                ;REPLACE ERROR CALL WITH
                                                ;"BR 4$" = 000764
033434                          6$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
033434  012737  033442  001110        MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
033442  012737  140340  177776  7$:   MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
033450  012700  036514                MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
033454  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
033460  006512                        MFPI    (R2)            ;READ FROM PHYSICAL 60000
033462  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
033464  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
033466  001401                        BEQ     8$              ;BRANCH IF CORRECT DATA WAS FETCHED
033470  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
                                                ;FOR TIGHTER SCOPE LOOP
                                                ;REPLACE ERROR CALL WITH
                                                ;"BR 7$" = 000764
033472                          8$:     ;THE FOLLOWING WILL TEST DSM=2 MFPI.
033472  012737  033500  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
033500  012737  140340  177776  9$:   MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
033506  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
033512  006522                        MFPI    (R2)+           ;READ FROM PHYSICAL 60000
033514  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
033516  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
033520  001401                        BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
033522  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
                                                ;FOR TIGHTER SCOPE LOOP
                                                ;REPLACE ERROR CALL WITH
                                                ;"BR 9$" = 000766
033524                          10$:    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
033524  012737  033532  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
033532  012737  140340  177776  11$:  MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
033540  006537  100000                MFPI    @#100000        ;READ FROM PHYSICAL 60000
033544  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
033546  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
033550  001401                        BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
033552  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
                                                ;FOR TIGHTER SCOPE LOOP
                                                ;REPLACE ERROR CALL WITH
                                                ;"BR 11$" = 000767
```

```
4008  033554                                12$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
4009  033554  012737  033562  001110                MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
4010  033562  012737  140340  177776  13$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE DERNEL PRESENT USER
4011  033570  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4012  033574  006542                        MFPI    -(R2)           ;READ FROM PHYSICAL 60000
4013  033576  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
4014  033600  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4015  033602  001401                        BEQ     14$             ;BRANCH IF CORRECT DATA WAS FETCHED
4016  033604  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
4017                                                                ;FOR TIGHTER SCOPE LOOP
4018                                                                ;REPLACE ERROR CALL WITH
4019                                                                ;"BR 13$" = 000766
4020  033606                                14$:    ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
4021                                                ;
4022  033606  012737  033614  001110                MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
4023  033614  012737  140340  177776  15$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4024  033622  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
4025  033630  012702  001204                MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
4026  033634  006552                        MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
4027  033636  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
4028  033640  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4029  033642  001401                        BEQ     16$             ;BRANCH IF CORRECT DATA FETCHED
4030  033644  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
4031                                                                ;FOR TIGHTER SCOPE LOOP
4032                                                                ;REPLACE ERROR CALL WITH
4033                                                                ;"BR 15$" = 000763
4034  033646                                16$:    ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
4035                                                ;
4036  033646  012737  033654  001110                MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$.
4037  033654  012737  140340  177776  17$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4038  033662  005002                        CLR     R2              ;MAKE REGISTER 2 A ZERO
4039  033664  006562  100000                MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
4040  033670  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
4041  033672  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4042  033674  001401                        BEQ     18$             ;BRANCH IF CORRECT DATA FETCHED
4043  033676  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
4044                                                                ;FOR TIGHTER SCOPE LOOP
4045                                                                ;REPLACE ERROR CALL WITH
4046                                                                ;"BR 17$" = 000766
4047  033700                                18$:    ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
4048                                                ;
4049  033700  012737  033706  001110                MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
4050  033706  012737  140340  177776  19$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4051  033714  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
4052  033722  012702  001202                MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
4053  033726  006572  000000                MFPI    @0(R2)          ;READ FROM PHYSICAL 60000
4054  033732  012601                        MOV     (USP)+,R1       ;POP USER STACK INTO R1
4055  033734  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4056  033736  001401                        BEQ     20$             ;BRANCH IF CORRECT DATA FETCHED
4057  033740  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
4058                                                                ;FOR TIGHTER SCOPE LOOP
4059                                                                ;REPLACE ERROR CALL WITH
4060                                                                ;"BR 19$" = 000762
4061  033742  012737  002314  000250  20$:    MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
4062  033750  012737  000340  177776          MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
4063  033756  012737  033272  001110          MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
```

```
 4064  033764  000423                          BR      TST51           ;;BRANCH TO NEXT TEXT
 4065
 4066
 4067  033766  012637  001356          21$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
 4068  033772  012637  001360                  MOV     (KSP)+,TRAPPS
 4069  033776  013737  177572  001362          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
 4070  034004  013737  177576  001364          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
 4071  034012  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
 4072  034020  104051                          ERROR   51              ;TRIED TO READ NON-RESIDENT PAGE
 4073                                                                  ;FOR TIGHTER SCOPE LOOP
 4074                                                                  ;REPLACE ERROR CALL WITH
 4075                                                                  ;A "NOP" = 000240
 4076  034022  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
 4077  034026  013746  001356                  MOV     TRAPPC,-(KSP)
 4078  034032  000002                          RTI                     ;RETURN TO TEST
 4079
 4080                                  ;;*****************************************************************
 4081                                  ;*TEST 51       MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
 4082                                  ;*
 4083                                  ;*      THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
 4084                                  ;*      BETWEEN INSTRUCTION AND DATA SPACE IN THE 11/34
 4085                                  ;*      MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI.
 4086                                  ;*
 4087                                  ;;*****************************************************************
 4088  034034  000004                  TST51:  SCOPE
 4089  034036  012737  030340  177776  1$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
 4090  034044  106506                          MFPD    USP             ;MFPD SHOULD ACT LIKE MFPI PUTTING
 4091                                                                  ;USER STACK POINTER ON THE KERNEL STACK
 4092  034046  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON KERNEL STACK?
 4093  034052  001407                          BEQ     2$              ;BRANCH IF NO
 4094  034054  012600                          MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
 4095  034056  012701  000700                  MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
 4096  034062  020001                          CMP     R0,R1           ;DID GET RIGHT POINTER VALUE?
 4097  034064  001403                          BEQ     3$              ;BRANCH IF YES
 4098  034066  104053                          ERROR   53              ;WRONG THING WAS PUSHED ON STACK
 4099                                                                  ;FOR TIGHTER SCOPE LOOP
 4100                                                                  ;REPLACE ERROR CALL WITH
 4101                                                                  ;"BR 1$" = 000763
 4102  034070  000401                          BR      3$              ;BRANCH TO NEXT TRY
 4103  034072  104054          2$:             ERROR   54              ;NOTHING PUSHED ON STACK
 4104                                                                  ;FOR TIGHTER SCOPE LOOP
 4105                                                                  ;REPLACE ERROR CALL WITH
 4106                                                                  ;"BR 1$" = 000761
 4107  034074  012737  034102  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
 4108  034102  012746  007777          4$:     MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
 4109  034106  106606                          MTPD    USP             ;LOAD THE USER STACK POINTER
 4110  034110  106506                          MFPD    USP             ;READ USER STACK POINTER
 4111  034112  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 4112  034114  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED?
 4113  034120  001401                          BEQ     5$              ;BRANCH IF YES
 4114  034122  104054                          ERROR   54              ;USER STACK POINTER NOT CHANGED
 4115                                                                  ;FOR TIGHTER SCOPE LOOP
 4116                                                                  ;REPLACE ERROR CALL WITH
 4117                                                                  ;"BR 4$" = 000767
 4118  034124  012746  000700          5$:     MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK. PTR.
 4119  034130  106606                          MTPD    USP             ;RESTORE USER STACK POINTER
```

```
 4120  034132  012737  034036  001110          MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
 4121
 4122                                  ;;*****************************************************************
 4123                                  ;*TEST 52        MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
 4124                                  ;*
 4125                                  ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
 4126                                  ;*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
 4127                                  ;*      STACK IS NOT DECREMENTED BEFORE ACCESS.
 4128                                  ;*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
 4129                                  ;*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
 4130                                  ;;*****************************************************************
 4131  034140  000004                  TST52:  SCOPE
 4132  034142  005037  177776          1$:     CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
 4133  034146  012700  001100                  MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
 4134  034152  010006                          MOV     R0,KSP          ;LOAD STACK POINTER
 4135  034154  006506                          MFPI    KSP             ;THE VALUE "STACK" SHOULD BE PUSHED
 4136                                                                  ;BEFORE BEING DECREMENTED
 4137  034156  011601                          MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
 4138  034160  020001                          CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE
 4139                                                                  ;STACK POINTER PUSHED?
 4140  034162  001401                          BEQ     2$              ;BRANCH IF YES
 4141  034164  104046                          ERROR   46              ;MFPI FETCHED WRONG DATA
 4142                                                                  ;FOR TIGHTER SCOPE LOOP
 4143                                                                  ;REPLACE ERROR CALL WITH
 4144                                                                  ;"BR 1$" = 000766
 4145  034166  005740                  2$:     TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
 4146  034170  020600                          CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
 4147  034172  001401                          BEQ     3$              ;BRANCH IF YES
 4148  034174  104050                          ERROR   50              ;STACK NOT PUSHED BY THE MFPI
 4149                                                                  ;FOR TIGHTER SCOPE LOOP
 4150                                                                  ;REPLACE ERROR CALL WITH
 4151                                                                  ;"BR 1$" = 000762
 4152  034176  012706  001100          3$:     MOV     #STACK,KSP      ;RESTORE STACK POINTER
 4153
 4154
 4155
 4156
 4157
 4158                          .SBTTL  ************************************************************
 4159
```

```
                                .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                ;*IF SW12=1 INHIBIT TRACE TRAP
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP

034202                          $EOP:
034202  000004                          SCOPE
034204  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
034210  005037  001212                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
034214  005237  001234                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
034220  042737  100000  001234          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
034226  005327                          DEC     (PC)+           ;;LOOP?
034230  000001          $EOPCT: .WORD   1
034232  003072                  BGT     $DOAGN          ;;YES
034234  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
034236  000001          $ENDCT: .WORD   1
034240  034230                  $EOPCT
034242  104401  034250          TYPE    ,65$            ;;TYPE ASCIZ STRING
034246  000407                  BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <12><15>/END PASS #/
034266                          64$:
034266  013746  001234          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                        ;;TYPE PASS NUMBER
034272  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
034274  104401  034302          TYPE    ,67$            ;;TYPE ASCIZ STRING
034300  000421                  BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
034344                          66$:
034344  013746  001112          MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                        ;;TOTAL NUMBER OF ERRORS
034350  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
034352  104401  001223          TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
034356  005037  001112          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
034362  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
034366  001414                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
034370  005046                  CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
034372  012746  034400          MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
034376  000426                  BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                        ;;WITH A CLEARED "T" BIT
034400                  $CLR.T:
034400  013700  000042          MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
034404  001405                  BEQ     $DOAGN          ;;RETURN ADDRESS
034406  000005                  RESET                   ;;CLEAR THE WORLD
034410  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
034412  000240                  NOP                     ;;SAVE ROOM
034414  000240                  NOP                     ;;FOR
034416  000240                  NOP                     ;;ACT11
034420                  $DOAGN:
034420  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
034422  042716  000020          BIC     #20,(SP)        ;;CLEAR THE "T" BIT
034426  032777  010000  144504  BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
```

```
4216  034434  001005                          BNE     1$              ;;BR IF NO
4217  034436  005137  034462                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
4218  034442  100402                          BMI     1$              ;;BR IF NO
4219  034444  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
4220  034450  012746  034456          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
4221  034454  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
4222                                                                  ;;AN "RTT" IF "RTT" IS A LEGAL
4223                                                                  ;;INSTRUCTION
4224  034456                          $LOOP:
4225  034456  000137                          JMP     @(PC)+          ;;RETURN
4226  034460  020474                  $RTNAD: .WORD   LOOP
4227  034462  000000                  $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
4228  034464     377      377      000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
4229            034470                  .EVEN
4230                                    .SBTTL  SCOPE HANDLER ROUTINE
4231
4232                                    ;;******************************************************************
4233                                    ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4234                                    ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4235                                    ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4236                                    ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4237                                    ;*SW14=1        LOOP ON TEST
4238                                    ;*SW11=1        INHIBIT ITERATIONS
4239                                    ;*SW09=1        LOOP ON ERROR
4240                                    ;*SW08=1        LOOP ON TEST IN SWR<7:0>
4241                                    ;*CALL
4242                                    ;*      SCOPE           ;;SCOPE=IOT
4243
4244  034470                          $SCOPE:
4245  034470  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
4246  034472  032777  040000  144440  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
4247  034500  001114                          BNE     $OVER           ;;YES IF SW14=1
4248                                    ;#####START OF CODE FOR THE XOR TESTER#####
4249  034502  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4250                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
4251  034504  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4252  034510  012737  034530  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
4253  034516  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
4254  034522  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4255  034526  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
4256  034530  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
4257  034532  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4258  034536  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
4259  034540                          6$:;#####END OF CODE FOR THE XOR TESTER#####
4260  034540  032777  000400  144372          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
4261  034546  001404                          BEQ     2$              ;;BR IF NO
4262  034550  127737  144364  001102          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
4263  034556  001465                          BEQ     $OVER           ;;BR IF YES
4264  034560  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
4265  034564  001421                          BEQ     3$              ;;BR IF NO
4266  034566  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4267  034574  101015                          BHI     3$              ;;BR IF NO
4268  034576  032777  001000  144334          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
4269  034604  001404                          BEQ     4$              ;;BR IF NO
4270  034606  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
4271  034614  000446                          BR      $OVER
```

```
4272  034616  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
4273  034622  005037  001212                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4274  034626  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
4275  034630  032777  004000  144302  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
4276  034636  001011                          BNE     1$              ;;BR IF YES
4277  034640  005737  001234                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
4278  034644  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
4279  034646  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
4280  034652  023737  001212  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
4281  034660  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
4282  034662  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
4283  034670  013737  034746  001212          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
4284  034676  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
4285  034702  113737  001102  001232          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
4286  034710  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
4287  034714  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
4288  034720  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4289  034724  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4290  034732  013777  001102  144202  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
4291  034740  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
4292  034744  000002                          RTI                     ;;FIXES PS
4293  034746  000200          $MXCNT: 200                             ;;MAX. NUMBER OF ITERATIONS
4294                          .SBTTL  ERROR HANDLER ROUTINE
4295
4296                          ;;**************************************************************
4297                          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4298                          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4299                          ;*AND GO TO ERRTYP ON ERROR
4300                          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4301                          ;*SW15=1         HALT ON ERROR
4302                          ;*SW13=1         INHIBIT ERROR TYPEOUTS
4303                          ;*SW10=1         BELL ON ERROR
4304                          ;*SW09=1         LOOP ON ERROR
4305                          ;*CALL
4306                          ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4307
4308  034750                  $ERROR:
4309  034750  104410                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
4310  034752  010037  001162          MOV     R0,$REG0                ;SAVE THE CONTENTS OF R0
4311  034756  010137  001164          MOV     R1,$REG1                ;SAVE THE CONTENTS OF R1
4312  034762  010237  001166          MOV     R2,$REG2                ;SAVE THE CONTENTS OF R2
4313  034766  010337  001170          MOV     R3,$REG3                ;SAVE THE CONTENTS OF R3
4314  034772  010437  001172          MOV     R4,$REG4                ;SAVE THE CONTENTS OF R4
4315  034776  010537  001174          MOV     R5,$REG5                ;SAVE THE CONTENTS OF R5
4316  035002  113737  001102  001352  MOVB    $TSTNM,TESTNO           ;SAVE THE TEST NUMBER
4317  035010  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
4318  035014  001775                  BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
4319  035016  013777  001102  144116  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4320  035024  032777  002000  144106  BIT     #BIT10,@SWR     ;;BELL ON ERROR?
4321  035032  001402                  BEQ     1$              ;;NO - SKIP
4322  035034  104401  001216          TYPE    ,$BELL          ;;RING BELL
4323  035040  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
4324  035044  011637  001116          MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
4325  035050  162737  000002  001116  SUB     #2,$ERRPC
4326  035056  117737  144034  001114  MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
4327  035064  032777  020000  144046  BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
```

```
4328  035072  001004                        BNE     20$             ;;SKIP TYPEOUTS
4329  035074  004737  035206                JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
4330  035100  104401  001223                TYPE    ,$CRLF
4331  035104                        20$:
4332  035104  122737  000001  001246        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4333  035112  001007                        BNE     2$              ;;NO,SKIP APT ERROR REPORT
4334  035114  113737  001114  035126        MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
4335  035122  004737  037540                JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
4336  035126     000                21$:    .BYTE   0
4337  035127     000                        .BYTE   0
4338  035130  000777                22$:    BR      22$             ;;APT ERROR LOOP
4339  035132  005777  144002        2$:     TST     @SWR            ;;HALT ON ERROR
4340  035136  100002                        BPL     3$              ;;SKIP IF CONTINUE
4341  035140  000000                        HALT                    ;;HALT ON ERROR!
4342  035142  104410                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
4343  035144  032777  001000  143766  3$:   BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
4344  035152  001402                        BEQ     4$              ;;BR IF NO
4345  035154  013716  001110                MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
4346  035160  005737  001214        4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
4347  035164  001402                        BEQ     5$              ;;BR IF NONE
4348  035166  013716  001214                MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
4349  035172                        5$:
4350  035172  022737  034410  000042        CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
4351  035200  001001                        BNE     6$              ;;BRANCH IF NO
4352  035202  000000                        HALT                    ;;YES
4353  035204                        6$:
4354  035204  000002                        RTI                     ;;RETURN
4355                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4356
4357                                ;;**************************************************************
4358                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4359                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4360                                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4361                                ;*
4362                                ;*NOTES:
4363                                ;*1)    THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
4364                                ;*      FOR "EM", "DH", AND "DT".
4365                                ;*2)    TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
4366                                ;*3)    FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4367                                ;*4)    THE AVAILABLE FORMATS FOR TYPING DATA ARE:
4368                                ;*      DF      FORMAT
4369                                ;*      0       TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
4370                                ;*      1       TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
4371                                ;*      2       TYPE A 16 DIGIT BINARY NUMBER
4372                                ;*      3       TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
4373                                ;*
4374
4375  035206                        ERRTYP:
4376  035206  104401  001223                TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
4377  035212  010046                        MOV     R0,-(SP)        ;SAVE R0
4378  035214  005000                        CLR     R0              ;PICKUP THE ITEM INDEX
4379  035216  153700  001114                BISB    @#$ITEMB,R0
4380  035222  001004                        BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
4381                                                                ;TYPE THE PC OF THE ERROR
4382  035224  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
4383                                                                ;;ERROR ADDRESS
```

```
 4384  035230  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 4385  035232  000471                        BR      13$             ;GET OUT
 4386  035234  005300                1$:     DEC     R0              ;ADJUST THE INDEX SO THAT IT WILL
 4387  035236  006300                        ASL     R0              ;       WORK FOR THE ERROR TABLE
 4388  035240  006300                        ASL     R0
 4389  035242  006300                        ASL     R0
 4390  035244  062700  001406              ADD     #$ERRTB,R0      ;FORM TABLE POINTER
 4391  035250  012037  035260              MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
 4392  035254  001404                      BEQ     3$              ;SKIP TYPEOUT IF NO POINTER
 4393  035256  104401                      TYPE                    ;TYPE THE "ERROR MESSAGE"
 4394  035260  000000                2$:     .WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
 4395  035262  104401  001223              TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
 4396  035266  012037  035276        3$:     MOV     (R0)+,4$        ;PICKUP "DATA HEADER" POINTER
 4397  035272  001404                      BEQ     5$              ;SKIP TYPEOUT IF 0
 4398  035274  104401                      TYPE                    ;TYPE THE "DATA HEADER"
 4399  035276  000000                4$:     .WORD   0               ;"DATA HEADER" POINTER GOES HERE
 4400  035300  104401  001223              TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
 4401  035304  010146                5$:     MOV     R1,-(SP)        ;SAVE R1
 4402  035306  012001                        MOV     (R0)+,R1        ;PICKUP "DATA TABLE" POINTER
 4403  035310  001441                        BEQ     12$             ;BR IF NO DATA TO BE TYPED
 4404  035312  012000                        MOV     (R0)+,R0        ;PICKUP "DATA FORMAT" POINTER
 4405  035314  105710                6$:     TSTB    (R0)            ;IS IT FORMAT 0?
 4406  035316  001003                        BNE     7$              ;BR IF NO
 4407
 4408                                  ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
 4409  035320  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
 4410  035322  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 4411  035324  000425                        BR      11$
 4412
 4413                                  ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
 4414  035326  121027  000001        7$:     CMPB    (R0),#1         ;IS IT FORMAT 1?
 4415  035332  001003                        BNE     8$              ;BRANCH IF NO
 4416  035334  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
 4417  035336  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
 4418  035340  000417                        BR      11$
 4419
 4420                                  ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
 4421  035342  121027  000002        8$:     CMPB    (R0),#2         ;IS IT FORMAT 2?
 4422  035346  001003                        BNE     9$              ;BRANCH IF NO
 4423  035350  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
 4424  035352  104406                        TYPBN                   ;;GO TYPE--BINARY ASCII
 4425  035354  000411                        BR      11$
 4426
 4427                                  ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
 4428  035356  012146                9$:     MOV     (R1)+,-(SP)     ;PUT ADDRESS OF FIRST LOC. ON STACK
 4429  035360  004737  040612              JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
 4430  035364  062716  000005              ADD     #5,(SP)         ;ONLY NEED 6 CHARACTERS NOT 11
 4431  035370  012637  035376              MOV     (SP)+,10$       ;PUT ADDRESS OF ASCII CHARS. AT 10$
 4432  035374  104401                      TYPE                    ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.
 4433  035376  000000                10$:    .WORD   0
 4434
 4435  035400  005711                11$:    TST     (R1)            ;IS THERE ANOTHER NUMBER?
 4436  035402  001404                      BEQ     12$             ;BR IF NO
 4437  035404  104401  035426              TYPE    ,14$            ;TYPE TWO(2) SPACES
 4438  035410  105720                      TSTB    (R0)+           ;POINT TO NEW "DATA FORMAT"
 4439  035412  000740                      BR      6$              ;LOOP
```

```
 4440
 4441  035414  012601                  12$:    MOV     (SP)+,R1        ;RESTORE R1
 4442  035416  012600                  13$:    MOV     (SP)+,R0        ;RESTORE R0
 4443  035420  104401  001223                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
 4444  035424  000207                          RTS     PC              ;RETURN
 4445  035426  020040      000         14$:    .ASCIZ  /  /            ;TWO(2) SPACES
 4446          035432                          .EVEN
 4447
```

```
                                .SBTTL  ***** SUBROUTINES USED BY THIS PROGRAM *****

                                .SBTTL  TURN OFF T-BIT AND SAVE CURRENT PSW
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
                                ;*      IF IT IS ON.  THE PROCESSOR STATUS IS SAVED IN "TBITPS" SO THAT
                                ;*      THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
                                ;*      WARRANT T-BIT TRAPPING.
                                ;*
                                ;;*****************************************************************
035432  033727  177776  000020  TOFF:   BIT     PSW,#TBIT       ;IS THE T-BIT SET IN THE PSW?
035440  001411                          BEQ     1$              ;EXIT IF NO
035442  013746  177776                  MOV     PSW,-(SP)       ;PUSH PRESENT PSW ON THE STACK
035446  011637  001366                  MOV     (SP),TBITPS     ;ALSO SAVE IT IN "TBITPS" FOR
                                                                ;RESTORING LATER
035452  042716  000020                  BIC     #TBIT,(SP)      ;CLEAR THE T-BIT (BIT 4) IN THE PSW
035456  012746  035464                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
035462  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT OFF
035464  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM

                                .SBTTL  TURN ON T-BIT AND RESTORE PREVIOUS PSW
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
                                ;*      PREVIOUS CONDITION BY RESTORING THE "T-BIT PSW" SAVED BY THE
                                ;*      "TOFF" SUBROUTINE IN THE "TBITPS" LOCATION.
                                ;*
                                ;;*****************************************************************
035466  033727  001366  000020  TON:    BIT     TBITPS,#TBIT    ;WAS T-BIT ON IN THE PREVIOUS PSW?
035474  001410                          BEQ     1$              ;EXIT IF NO
035476  013746  001366                  MOV     TBITPS,-(SP)    ;PUSH PREVIOUS PSW ON THE STACK
035502  012737  000340  001366          MOV     #340,TBITPS     ;RESET THE "TBITPS" LOCATION
035510  012746  035516                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
035514  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT RESTORED
035516  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM

                                .SBTTL  SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
                                ;*      TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
                                ;*      PDR'S TO A 1.  THE "INITIAL STATE" OF HAVING ALL BITS=1 IS
                                ;*      USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
                                ;*      A SINGLE PAR OR PDR ADDRESS.
                                ;*
                                ;;*****************************************************************
035520  012702  000010  SETREG: MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
035524  012701  172300          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
035530  012721  177777  1$:     MOV     #-1,(R1)+       ;SET BITS IN KERNEL PDR TO 1
035534  077203                  SOB     R2,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
035536  012702  000010          MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
035542  012701  172340          MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
035546  012721  177777  2$:     MOV     #-1,(R1)+       ;SET BITS IN A KERNEL PAR TO 1
035552  077203                  SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

```
 4504  035554  012702  000010            MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
 4505  035560  012701  177600            MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
 4506  035564  012721  177777  3$:       MOV     #-1,(R1)+       ;SET BITS IN A USER PDR TO 1
 4507  035570  077203                    SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
 4508  035572  012702  000010            MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
 4509  035576  012701  177640            MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
 4510  035602  012721  177777  4$:       MOV     #-1,(R1)+       ;SET BITS IN A USER PAR TO 1
 4511  035606  077203                    SOB     R2,4$           ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
 4512  035610  000207                    RTS     PC              ;RETURN TO TEST
 4513
 4514                          .SBTTL  READ & COMPARE KERNEL & USER PAR/PDR'S
 4515                          ;;*****************************************************************
 4516                          ;*
 4517                          ;*      THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
 4518                          ;*      READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
 4519                          ;*      WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
 4520                          ;*      ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
 4521                          ;*      BE REPORTED BY THIS SUBROUTINE.
 4522                          ;*
 4523                          ;;*****************************************************************
 4524  035612                  CMPREG:
 4525  035612  012701  172300            MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
 4526  035616  012704  000010            MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4527  035622  012705  077416            MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
 4528  035626  011102  1$:       MOV     (R1),R2         ;READ A KERNEL PDR INTO R2
 4529  035630  020205                    CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AL EXPECTED?
 4530  035632  001403                    BEQ     2$              ;BRANCH IF YES
 4531  035634  020100                    CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
 4532  035636  001401                    BEQ     2$              ;BRANCH IF YES
 4533  035640  104016                    ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
 4534                                                          ;FOR TIGHTER SCOPE LOOP
 4535                                                          ;REPLACE ERROR CALL WITH
 4536                                                          ;AN "RTS PC" = 000207
 4537  035642  062701  000002  2$:       ADD     #2,R1           ;FORM NEXT KERNEL PDR ADDRESS
 4538  035646  077411                    SOB     R4,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
 4539  035650  012701  172340            MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
 4540  035654  012704  000010            MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4541  035660  012705  007777            MOV     #7777,R5        ;PUT EXPECTED PAR CONTENTS IN R5
 4542  035664  011102  3$:       MOV     (R1),R2         ;READ A KERNEL PAR INTO R2
 4543  035666  020205                    CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
 4544  035670  001403                    BEQ     4$              ;BRANCH IF YES
 4545  035672  020100                    CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
 4546  035674  001401                    BEQ     4$              ;BRANCH IF YES
 4547  035676  104016                    ERROR   16              ;A PAR WAS EFFECTED BY CLEARING A DIFFENENT PAR/PDR
 4548                                                          ;FOR TIGHTER SCOPE LOOP
 4549                                                          ;REPLACE ERROR CALL WITH
 4550                                                          ;AN "RTS PC" = 000207
 4551  035700  062701  000002  4$:       ADD     #2,R1           ;FORM NEXT KERNEL PAR ADDRESS
 4552  035704  077411                    SOB     R4,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
 4553  035706  012701  177600            MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST USER PDR IN R1
 4554  035712  012704  000010            MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4555  035716  012705  077416            MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
 4556  035722  011102  5$:       MOV     (R1),R2         ;READ A USER PDR INTO R2
 4557  035724  020205                    CMP     R2,R5           ;ARE ALL WRITABLE BITS SET AS EXPECTED?
 4558  035726  001403                    BEQ     6$              ;BRANCH IF YES
 4559  035730  020100                    CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
```

```
4560  035732  001401                        BEQ     6$              ;BRANCH IF YES
4561  035734  104016                        ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4562                                                                ;FOR TIGHTER SCOPE LOOP
4563                                                                ;REPLACE ERROR CALL WITH
4564                                                                ;AN "RTS PC" = 000207
4565  035736  062701  000002          6$:     ADD     #2,R1           ;FORM NEXT USER PDR ADDRESS
4566  035742  077411                        SOB     R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4567  035744  012701  177640                MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
4568  035750  012704  000010                MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4569  035754  012705  007777                MOV     #7777,R5        ;PUT EXPECTED PAR CONTENTS IN R5
4570  035760  011102                  7$:     MOV     (R1),R2         ;READ A USER PAR INTO R2
4571  035762  020205                        CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4572  035764  001403                        BEQ     8$              ;BRANCH IF YES
4573  035766  020100                        CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4574  035770  001401                        BEQ     8$              ;BRANCH IF YES
4575  035772  104016                        ERROR   16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4576                                                                ;FOR TIGHTER SCOPE LOOP
4577                                                                ;REPLACE ERROR CALL WITH
4578                                                                ;AN "RTS PC" = 000207
4579  035774  062701  000002          8$:     ADD     #2,R1           ;FORM NEXT USER PAR ADDRESS
4580  036000  077411                        SOB     R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4581  036002  000207                        RTS     PC              ;RETURN TO TEST
4582
4583                                  .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
4584                                  ;;******************************************************************
4585                                  ;*
4586                                  ;*      THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
4587                                  ;*      (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
4588                                  ;*      PAGE ADDRESS REGISTER (PAR).  THE SAME METHOD USED BY THE MEMORY
4589                                  ;*      MANAGEMENT LOGIC IS USED.  VBA <15:13> SELECTS WHICH PAR/PDR
4590                                  ;*      IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
4591                                  ;*      TO PAR <11:00> TO GIVE PBA <17:6>.  BITS <17:16> OF THE
4592                                  ;*      PHYSICAL ADDRESS ARE LEFT IN LOC. "PBAHI" AND BITS <15:00>
4593                                  ;*      ARE LEFT IN LOC. "PBALO".  THE PSW'S "CURRENT MODE" BITS
4594                                  ;*      ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S.  THE ROUTINE
4595                                  ;*      IS ENTERED WITH LOC. "VIRT1" CONTAINING THE 16-BIT VIRTUAL
4596                                  ;*      ADDRESS.
4597                                  ;*
4598                                  ;;******************************************************************
4599
4600  036004  012702  172340          FORMPA: MOV     #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4601  036010  032737  140000  177776          BIT     #140000,PSW     ;IN USER MODE?
4602  036016  001402                        BEQ     1$              ;BRANCH IF NO
4603  036020  012702  177640                MOV     #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
4604  036024  013700  001376          1$:     MOV     VIRT1,R0        ;LOAD VIRTUAL ADDR. (VBA) INTO R0
4605  036030  072027  177764                ASH     #-14,R0         ;GET BITS <15:13> DOWN TO BITS <3:1>
4606  036034  042700  177761                BIC     #177761,R0      ;MASK OF ALL BITS BUT BITS <3:1>
4607  036040  060002                        ADD     R0,R2           ;ADD OFFSET TO BASE PAR ADDRESS
4608  036042  011200                        MOV     (R2),R0         ;GET BITS <11:00> FROM APPROPRIATE PAR
4609  036044  010002                        MOV     R0,R2           ;COPY PAR BITS <11:00> INTO R2
4610  036046  013737  001376  001402        MOV     VIRT1,PBALO     ;PUT VIRTUAL ADDR. IN LOC. "PBALO"
4611  036054  042737  160000  001402        BIC     #160000,PBALO   ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4612  036062  072227  177766                ASH     #-12,R2         ;GET PAR <11:00> DOWN TO BITS <1:0> OF R2
4613  036066  042702  177774                BIC     #177774,R2      ;CLEAR OFF ALL BITS BUT BITS <1:0>
4614  036072  072027  000006                ASH     #6,R0           ;SHIFT PAR<9:0> TO <15:6> OF R0
4615  036076  042700  000077                BIC     #77,R0          ;CLEAR BITS <5:0> OF R0
```

```
4616  036102  060037  001402          ADD     R0,PBALO        ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
4617                                                          ;(PAR<9:0> IN BITS <15:6> OF R0)
4618  036106  005502                  ADC     R2              ;ADD ANY CARRY TO R2
4619  036110  010237  001404          MOV     R2,PBAHI        ;PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
4620  036114  000207                  RTS     PC              ;RETURN TO PROGRAM
4621
```

```
                                .SBTTL  TTY INPUT ROUTINE
                                ;;*****************************************************************
                                .ENABL  LSB
                                ;;*****************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
036116  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
036124  001114                         BNE     15$             ;;BRANCH IF NO
036126  105777  143012                 TSTB    @$TKS           ;;CHAR THERE?
036132  100111                         BPL     15$             ;;IF NO, DON'T WAIT AROUND
036134  117746  143006                 MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
036140  042716  177600                 BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
036144  022726  000007                 CMP     #7,(SP)+        ;;IS IT A CONTROL G?
036150  001102                         BNE     15$             ;;NO, RETURN TO USER
036152  123727  001134  000001         CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
036160  001476                         BEQ     15$             ;;BRANCH IF YES
                                
036162  104401  037060                 TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
036166  104401  037065         $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
036172  013746  000176                 MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
036176  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
036200  104401  037076                 TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
036204  005046                 19$:    CLR     -(SP)           ;;CLEAR COUNTER
036206  005046                         CLR     -(SP)           ;;THE NEW SWR
036210  105777  142730         7$:     TSTB    @$TKS           ;;CHAR THERE?
036214  100375                         BPL     7$              ;;IF NOT TRY AGAIN
                                
036216  117746  142724                 MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
036222  042716  177600                 BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
                                
036226  021627  000003                 CMP     (SP),#3         ;;IS IT A CONTROL-C?
036232  001015                         BNE     9$              ;;BRANCH IF NOT
036234  104401  037046                 TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
036240  062706  000006                 ADD     #6,SP           ;;CLEAN UP STACK
036244  123727  001135  000001         CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
036252  001003                         BNE     8$              ;;BRANCH IF NO
036254  012777  000100  142662         MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
036262  000137  037110         8$:     JMP     CNTRLC          ;;CONTROL-C RESTART
                                
036266  021627  000025         9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
036272  001005                         BNE     10$             ;;BRANCH IF NOT
036274  104401  037053                 TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
036300  062706  000006         20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
036304  000737                         BR      19$             ;;LET'S TRY IT AGAIN
                                
036306  021627  000015         10$:    CMP     (SP),#15        ;;IS IT A <CR>?
036312  001022                         BNE     16$             ;;BRANCH IF NO
036314  005766  000004                 TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
036320  001403                         BEQ     11$             ;;BRANCH IF YES
036322  016677  000002  142610         MOV     2(SP),@SWR      ;;SAVE NEW SWR
```

```
4678  036330  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
4679  036334  104401  001223          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
4680  036340  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
4681  036346  001003                          BNE     15$             ;;BRANCH IF NOT
4682  036350  012777  000100  142566          MOV     #100,@STKS      ;;RE-ENABLE TTY KBD INTERRUPTS
4683  036356  000002                  15$:    RTI                     ;;RETURN
4684  036360  004737  037452          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
4685  036364  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
4686  036370  002420                          BLT     18$             ;;BRANCH IF YES
4687  036372  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
4688  036376  003015                          BGT     18$             ;;BRANCH IF YES
4689  036400  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
4690  036404  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
4691  036410  001403                          BEQ     17$             ;;BRANCH IF YES
4692  036412  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
4693  036414  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
4694  036416  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
4695  036420  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
4696  036424  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
4697  036430  000667                          BR      7$              ;;GET THE NEXT ONE
4698  036432  104401  001222          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
4699  036436  000720                          BR      20$             ;;SIMULATE CONTROL-U
4700                                  .DSABL  LSB
4701
4702
4703                                  ;;*****************************************************************
4704                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4705                                  ;*CALL:
4706                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4707                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4708                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
4709                                  ;
4710
4711  036440  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4712  036442  016666  000004  000002  MOV     4(SP),2(SP)     ;;SAVE THE PS
4713  036450  105777  142470          1$:     TSTB    @$TKS           ;;WAIT FOR
4714  036454  100375                          BPL     1$              ;;A CHARACTER
4715  036456  117766  142464  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
4716  036464  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4717  036472  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
4718  036500  001013                          BNE     3$              ;;BRANCH IF NO
4719  036502  105777  142436          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
4720  036506  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
4721  036510  117746  142432                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
4722  036514  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
4723  036520  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
4724  036524  001366                          BNE     2$              ;;IF NOT DISCARD IT
4725  036526  000750                          BR      1$              ;;YES, RESUME
4726  036530  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
4727  036536  002407                          BLT     4$              ;;BRANCH IF YES
4728  036540  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
4729  036546  003003                          BGT     4$              ;;BRANCH IF YES
4730  036550  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
4731  036556  000002                  4$:     RTI                     ;;GO BACK TO USER
4732                                  ;;*****************************************************************
4733                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
```

```
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
036560  010346                  $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
036562  005046                          CLR     -(SP)                   ;;CLEAR THE RUBOUT KEY
036564  012703  037036          1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
036570  022703  037046          2$:     CMP     #$TTYIN+80.,R3          ;;BUFFER FULL?
036574  101467                          BLOS    4$                      ;;BR IF YES
036576  104411                          RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
036600  112613                          MOVB    (SP)+,(R3)              ;;GET CHARACTER
036602  122713  000003                  CMPB    #3,(R3)                 ;;IS IT A CONTROL-C?
036606  001006                          BNE     10$                     ;;BRANCH IF NO
036610  104401  037046                  TYPE    ,$CNTLC                 ;;TYPE A CONTROL-C (↑C)
036614  005726                          TST     (SP)+                   ;;CLEAN RUBOUT KEY OFF OF THE STACK
036616  012603                          MOV     (SP)+,R3                ;;RESTORE R3
036620  000137  037110                  JMP     CNTRLC                  ;;GOTO CONTROL-C RESTART
036624  122713  000177          10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
036630  001022                          BNE     5$                      ;;BR IF NO
036632  005716                          TST     (SP)                    ;;IS THIS THE FIRST RUBOUT?
036634  001007                          BNE     6$                      ;;BR IF NO
036636  112737  000134  037034        MOVB    #'\,9$                  ;;TYPE A BACK SLASH
036644  104401  037034                  TYPE    ,9$
036650  012716  177777                  MOV     #-1,(SP)                ;;SET THE RUBOUT KEY
036654  005303                  6$:     DEC     R3                      ;;BACKUP BY ONE
036656  020327  037036                  CMP     R3,#$TTYIN              ;;STACK EMPTY?
036662  103434                          BLO     4$                      ;;BR IF YES
036664  111337  037034                  MOVB    (R3),9$                 ;;SETUP TO TYPEOUT THE DELETED CHAR.
036670  104401  037034                  TYPE    ,9$                     ;;GO TYPE
036674  000735                          BR      2$                      ;;GO READ ANOTHER CHAR.
036676  005716                  5$:     TST     (SP)                    ;;RUBOUT KEY SET?
036700  001406                          BEQ     7$                      ;;BR IF NO
036702  112737  000134  037034        MOVB    #'\,9$                  ;;TYPE A BACK SLASH
036710  104401  037034                  TYPE    ,9$
036714  005016                          CLR     (SP)                    ;;CLEAR THE RUBOUT KEY
036716  122713  000025          7$:     CMPB    #25,(R3)                ;;IS CHARACTER A CTRL U?
036722  001003                          BNE     8$                      ;;BR IF NO
036724  104401  037053                  TYPE    ,$CNTLU                 ;;TYPE A CONTROL "U"
036730  000715                          BR      1$                      ;;GO START OVER
036732  122713  000022          8$:     CMPB    #22,(R3)                ;;IS CHARACTER A "↑R"?
036736  001011                          BNE     3$                      ;;BRANCH IF NO
036740  105013                          CLRB    (R3)                    ;;CLEAR THE CHARACTER
036742  104401  001223                  TYPE    ,$CRLF                  ;;TYPE A "CR" & "LF"
036746  104401  037036                  TYPE    ,$TTYIN                 ;;TYPE THE INPUT STRING
036752  000706                          BR      2$                      ;;GO PICKUP ANOTHER CHACTER
036754  104401  001222          4$:     TYPE    ,$QUES                  ;;TYPE A '?'
036760  000701                          BR      1$                      ;;CLEAR THE BUFFER AND LOOP
036762  111337  037034          3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
036766  104401  037034                  TYPE    ,9$
036772  122723  000015                  CMPB    #15,(R3)+               ;;CHECK FOR RETURN
036776  001274                          BNE     2$                      ;;LOOP IF NOT RETURN
037000  105063  177777                  CLRB    -1(R3)                  ;;CLEAR RETURN (THE 15)
037004  104401  001224                  TYPE    ,$LF                    ;;TYPE A LINE FEED
037010  005726                          TST     (SP)+                   ;;CLEAN RUBOUT KEY FROM THE STACK
037012  012603                          MOV     (SP)+,R3                ;;RESTORE R3
```

```
 4790  037014  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
 4791  037016  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
 4792  037024  012766  037036  000004        MOV     #$TTYIN,4(SP)
 4793  037032  000002                        RTI                     ;;RETURN
 4794  037034     000                  9$:    .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
 4795  037035     000                          .BYTE   0               ;;TERMINATOR
 4796  037036  000010               $TTYIN:  .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
 4797  037046  041536  005015     000  $CNTLC:  .ASCIZ  /↑C/<15><12>   ;;CONTROL "C"
 4798  037053     136  006525  000012  $CNTLU:  .ASCIZ  /↑U/<15><12>   ;;CONTROL "U"
 4799  037060  043536  005015     000  $CNTLG:  .ASCIZ  /↑G/<15><12>   ;;CONTROL "G"
 4800  037065     015  051412  051127  $MSWR:   .ASCIZ  <15><12>/SWR = /
 4801  037072  036440  000040
 4802  037076  020040  042516  020127  $MNEW:   .ASCIZ  /  NEW = /
 4803  037104  020075     000
 4804          037110                  .EVEN
 4805
 4806                                  .SBTTL  CONTROL-C SERVICING ROUTINE
 4807
 4808                                  ;*      THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
 4809                                  ;*      BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
```

```
                                        ;*      (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
                                        ;*      A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
                                        ;*      THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
                                        ;*      AND THEN THE PROGRAM WILL GO TO "END-OF-PASS" AND CONTINUE
                                        ;*
037110  013737  001234  001210  CNTRLC: MOV     $PASS,$TMP5        ;GET THE VALUE OF "$PASS"
037116  005237  001210                  INC     $TMP5              ;FORM CURRENT PASS NO.
037122  104401  037167                  TYPE    ,CMSG              ;TYPE THE TEST STOPS MESSAGE
037126  113737  001102  037162          MOVB    $TSTNM,1$          ;SAVE THE TEST NUMBER
037134  013746  037162                  MOV     1$,-(SP)           ;;SAVE 1$ FOR TYPEOUT
037140  104402                          TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
037142  104401  037164                  TYPE    ,2$                ;TYPE 2 SPACES
037146  013746  001210                  MOV     $TMP5,-(SP)        ;;SAVE $TMP5 FOR TYPEOUT
037152  104405                          TYPDS                      ;;GO TYPE--DECIMAL ASCII WITH SIGN
037154  104407                          GTSWR                      ;ASK FOR NEW SWR VALUE
037156  000137  034204                  JMP     $EOP+2             ;CONTINUE AT $EOP+2
037162  000000                  1$:     .WORD   0                  ;BUFFER FOR TEST NUMBER
037164  020040      000         2$:     .ASCIZ  /  /               ;TWO SPACES AND THE STOP MESSAGE
037167     112  046525  044520  CMSG:   .ASCII  /JUMPING TO END-OF-PASS/<15><12>
037174  043516  052040  020117
037202  047105  026504  043117
037210  050055  051501  006523
037216     012
037217     124  051505  047124          .ASCIZ  /TESTNO  PASSNO/<15><12>
037224  020117  050040  051501
037232  047123  006517  000012
                                        .EVEN
                                .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
                                        ;*
037240  105737  001157          $TYPE:  TSTB    $TPFLG             ;;IS THERE A TERMINAL?
037244  100002                          BPL     1$                 ;;BR IF YES
037246  000000                          HALT                       ;;HALT HERE IF NO TERMINAL
037250  000430                          BR      3$                 ;;LEAVE
037252  010046                  1$:     MOV     R0,-(SP)           ;;SAVE R0
037254  017600  000002                  MOV     @2(SP),R0          ;;GET ADDRESS OF ASCIZ STRING
037260  122737  000001  001246          CMPB    #APTENV,$ENV       ;;RUNNING IN APT MODE
037266  001011                          BNE     62$                ;;NO,GO CHECK FOR APT CONSOLE
037270  132737  000100  001247          BITB    #APTSPOOL,$ENVM    ;;SPOOL MESSAGE TO APT
037276  001405                          BEQ     62$                ;;NO,GO CHECK FOR CONSOLE
037300  010037  037310                  MOV     R0,61$             ;;SETUP MESSAGE ADDRESS FOR APT
037304  004737  037530                  JSR     PC,$ATY3           ;;SPOOL MESSAGE TO APT
```

```
4866  037310  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
4867  037312  132737  000040  001247  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
4868  037320  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
4869  037322  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4870  037324  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
4871  037326  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
4872  037330  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
4873  037332  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
4874  037336  000002                          RTI                     ;;RETURN
4875  037340  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
4876  037344  001430                          BEQ     8$
4877  037346  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
4878  037352  001006                          BNE     5$
4879  037354  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
4880  037356  104401                          TYPE                    ;;TYPE A CR AND LF
4881  037360  001223                          $CRLF
4882  037362  105037  037516                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
4883  037366  000755                          BR      2$              ;;GET NEXT CHARACTER
4884  037370  004737  037452          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
4885  037374  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
4886  037400  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
4887  037402  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
4888                                                                  ;;AND THE NULL CHAR.
4889  037406  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
4890  037412  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
4891  037414  004737  037452                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
4892  037420  105337  037516                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
4893  037424  000770                          BR      7$              ;;LOOP
4894
4895                                  ;HORIZONTAL TAB PROCESSOR
4896
4897  037426  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
4898  037432  004737  037452          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
4899  037436  132737  000007  037516          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
4900  037444  001372                          BNE     9$              ;;TAB STOP
4901  037446  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
4902  037450  000724                          BR      2$              ;;GET NEXT CHARACTER
4903  037452  105777  141472          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
4904  037456  100375                          BPL     $TYPEC
4905  037460  116677  000002  141464          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4906  037466  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
4907  037474  001003                          BNE     1$              ;;BRANCH IF NO
4908  037476  105037  037516                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
4909  037502  000406                          BR      $TYPEX          ;;EXIT
4910  037504  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
4911  037512  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
4912  037514  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
4913  037516  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
4914  037520  000207          $TYPEX: RTS     PC
4915
4916                                  .SBTTL  APT COMMUNICATIONS ROUTINE
4917
4918                                  ;;**************************************************************************
4919  037522  112737  000001  037766  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
4920  037530  112737  000001  037764  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
4921  037536  000403                          BR      $ATYC
```

```
4922  037540  112737  000001  037766  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
4923  037546                          $ATYC:
4924  037546  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4925  037550  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4926  037552  105737  037764                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
4927  037556  001450                          BEQ     5$              ;;IF NOT:  BR
4928  037560  122737  000001  001246          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
4929  037566  001031                          BNE     3$              ;;IF NOT:  BR
4930  037570  132737  000100  001247          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4931  037576  001425                          BEQ     3$              ;;IF NOT:  BR
4932  037600  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
4933  037604  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4934  037612  005737  001226          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
4935  037616  001375                          BNE     1$              ;;IF NOT:  WAIT
4936  037620  010037  001242                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
4937  037624  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
4938  037626  001376                          BNE     2$
4939  037630  163700  001242                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
4940  037634  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
4941  037636  010037  001244                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
4942  037642  012737  000004  001226          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
4943  037650  000413                          BR      5$
4944  037652  017637  000004  037676  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
4945  037660  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
4946  037666  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
4947  037672  004737  037240                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
4948  037676  000000                  4$:     .WORD   0
4949  037700                          5$:
4950  037700  105737  037766          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
4951  037704  001416                          BEQ     12$             ;;IF NOT:  BR
4952  037706  005737  001246                  TST     $ENV            ;;RUNNING UNDER APT?
4953  037712  001413                          BEQ     12$             ;;IF NOT:  BR
4954  037714  005737  001226          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
4955  037720  001375                          BNE     11$             ;;IF NOT:  WAIT
4956  037722  017637  000004  001230          MOV     @4(SP),$FATAL   ;;GET ERROR #
4957  037730  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4958  037736  005237  001226                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
4959  037742  105037  037766          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
4960  037746  105037  037765                  CLRB    $LFLG           ;;CLEAR LOG FLAG
4961  037752  105037  037764                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
4962  037756  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4963  037760  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4964  037762  000207                          RTS     PC              ;;RETURN
4965  037764     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
4966  037765     000          $LFLG:  .BYTE   0               ;;LOG FLAG
4967  037766     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
4968        037770                  .EVEN
4969        000200          APTSIZE=200
4970        000001          APTENV=001
4971        000100          APTSPOOL=100
4972        000040          APTCSUP=040
4973                        .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
4974
4975                        ;;******************************************************************
4976                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
4977                        ;*BINARY-ASCII NUMBER AND TYPE IT.
```

```
                                        ;*CALL:
                                        ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
                                        ;*      TYPBN                   ;;TYPE IT
                                        ;*
037770  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
037772  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
037776  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
040000  112737  000060  040042  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
040006  006101                          ROL     R1              ;;GET THIS BIT
040010  001406                          BEQ     2$              ;;DONE?
040012  105537  040042                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
040016  104401  040042                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
040022  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
040024  000765                          BR      1$              ;;GO DO THE NEXT BIT
040026  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
040030  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
040036  012616                          MOV     (SP)+,(SP)
040040  000002                          RTI                     ;;RETURN TO USER
040042    000     000           $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

040044  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
040050  116637  000001  040267          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
040056  112637  040271                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
040062  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
040066  000406                          BR      $TYPON
040070  112737  000001  040267  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
040076  112737  000006  040271          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
040104  112737  000005  040266  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
040112  010346                          MOV     R3,-(SP)        ;;SAVE R3
040114  010446                          MOV     R4,-(SP)        ;;SAVE R4
040116  010546                          MOV     R5,-(SP)        ;;SAVE R5
040120  113704  040271                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
```

```
5034  040124  005404                        NEG     R4
5035  040126  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
5036  040132  110437  040270                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
5037  040136  113704  040267                MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
5038  040142  016605  000012                MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
5039  040146  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
5040  040150  006105                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
5041  040152  000404                        BR      3$              ;;GO DO MSB
5042  040154  006105                2$:     ROL     R5              ;;FORM THIS DIGIT
5043  040156  006105                        ROL     R5
5044  040160  006105                        ROL     R5
5045  040162  010503                        MOV     R5,R3
5046  040164  006103                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
5047  040166  105337  040270                DECB    $OMODE          ;;TYPE THIS DIGIT?
5048  040172  100016                        BPL     7$              ;;BR IF NO
5049  040174  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
5050  040200  001002                        BNE     4$              ;;TEST FOR 0
5051  040202  005704                        TST     R4              ;;SUPPRESS THIS 0?
5052  040204  001403                        BEQ     5$              ;;BR IF YES
5053  040206  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
5054  040210  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
5055  040214  052703  000040        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
5056  040220  110337  040264                MOVB    R3,8$           ;;SAVE FOR TYPING
5057  040224  104401  040264                TYPE    ,8$             ;;GO TYPE THIS DIGIT
5058  040230  105337  040266        7$:     DECB    $OCNT           ;;COUNT BY 1
5059  040234  003347                        BGT     2$              ;;BR IF MORE TO DO
5060  040236  002402                        BLT     6$              ;;BR IF DONE
5061  040240  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5062  040242  000744                        BR      2$              ;;GO DO THE LAST DIGIT
5063  040244  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
5064  040246  012604                        MOV     (SP)+,R4        ;;RESTORE R4
5065  040250  012603                        MOV     (SP)+,R3        ;;RESTORE R3
5066  040252  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
5067  040260  012616                        MOV     (SP)+,(SP)
5068  040262  000002                        RTI                     ;;RETURN
5069  040264     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
5070  040265     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
5071  040266     000                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
5072  040267     000                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
5073  040270  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
5074                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5075
5076                                ;;**************************************************************
5077                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5078                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5079                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5080                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5081                                ;*REPLACED WITH SPACES.
5082                                ;*CALL:
5083                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
5084                                ;*      TYPDS                   ;;GO TO THE ROUTINE
5085
5086  040272                        $TYPDS:
5087  040272  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5088  040274  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5089  040276  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

```
5090  040300  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5091  040302  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5092  040304  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
5093  040310  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
5094  040314  100004                        BPL     1$              ;;BR IF INPUT IS POS.
5095  040316  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
5096  040320  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
5097  040326  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
5098  040330  012703  040506                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
5099  040334  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
5100  040340  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
5101  040342  016001  040476                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
5102  040346  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
5103  040350  002402                        BLT     4$              ;;BR IF DONE
5104  040352  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
5105  040354  000774                        BR      3$
5106  040356  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
5107  040360  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
5108  040362  001002                        BNE     5$              ;;FALL THROUGH IF 0
5109  040364  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
5110  040366  100407                        BMI     7$              ;;BR IF YES
5111  040370  106316                5$:     ASLB    (SP)            ;;MSD?
5112  040372  103003                        BCC     6$              ;;BR IF NO
5113  040374  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
5114  040402  052702  000060        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
5115  040406  052702  000040        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5116  040412  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5117  040414  005720                        TST     (R0)+           ;;JUST INCREMENTING
5118  040416  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
5119  040422  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
5120  040424  003002                        BGT     8$              ;;GO TO EXIT
5121  040426  010502                        MOV     R5,R2           ;;GET THE LSD
5122  040430  000764                        BR      6$              ;;GO CHANGE TO ASCII
5123  040432  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
5124  040434  100003                        BPL     9$              ;;BR IF NO
5125  040436  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
5126  040444  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
5127  040446  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
5128  040450  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
5129  040452  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
5130  040454  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
5131  040456  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
5132  040460  104401  040506                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
5133  040464  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
5134  040472  012616                        MOV     (SP)+,(SP)
5135  040474  000002                        RTI                     ;;RETURN TO USER
5136  040476  023420                $DTBL:  10000.
5137  040500  001750                        1000.
5138  040502  000144                        100.
5139  040504  000012                        10.
5140  040506  000004                $DBLK:  .BLKW   4
5141                                  .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5142
5143                                  ;;******************************************************************
5144                                  ;*SAVE R0-R5
5145                                  ;*CALL:
```

```
                                ;*      SAVREG
                                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                ;*
                                ;*TOP---(+16)
                                ;*  +2---(+18)
                                ;*  +4---R5
                                ;*  +6---R4
                                ;*  +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0
                                ;
040516                          $SAVREG:
040516  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
040520  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
040522  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
040524  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
040526  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
040530  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
040532  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
040536  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
040542  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
040546  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
040552  000002                          RTI
                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
040554                          $RESREG:
040554  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
040560  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
040564  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
040570  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
040574  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
040576  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
040600  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
040602  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
040604  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
040606  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
040610  000002                          RTI
                                .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
                                ;;
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
                                ;
040612  104413                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
040614  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
040620  012705  040731                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
040624  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
040630  012703  177770                  MOV     #^C7,R3         ;;MASK
```

```
5202  040634  012100                  MOV     (R1)+,R0        ;;LOWER WORD
5203  040636  012101                  MOV     (R1)+,R1        ;;HIGH WORD
5204  040640  005002                  CLR     R2              ;;TERMINATOR
5205  040642  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
5206  040644  010002                  MOV     R0,R2           ;;GET THIS DIGIT
5207  040646  005304                  DEC     R4              ;;COUNT THIS CHARACTER
5208  040650  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
5209  040652  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
5210  040654  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5211  040656  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
5212  040662  104414                  RESREG                  ;;RESTORE ALL REGISTERS
5213  040664  000207                  RTS     PC              ;;RETURN TO USER
5214  040666  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
5215  040670  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
5216  040672  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
5217  040674  006001                  ROR     R1
5218  040676  006000                  ROR     R0
5219  040700  006001                  ROR     R1
5220  040702  006000                  ROR     R0
5221  040704  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
5222  040706  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
5223  040712  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
5224  040714  000016          $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
                                .SBTTL  TRAP DECODER

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                ;*GO TO THAT ROUTINE.

040732  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
040734  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
040740  005740                          TST     -(R0)           ;;BACKUP BY 2
040742  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
040744  006300                          ASL     R0              ;;POSITION FOR INDEXING
040746  016000  040766                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
040752  000200                          RTS     R0              ;;GO TO ROUTINE

                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

040754  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
040756  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
040764  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.

                                ;       ROUTINE
                                ;       -------
040766  040754                  $TRPAD: .WORD   $TRAP2
040770  037240                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
040772  040070                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
040774  040044                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
040776  040104                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
041000  040272                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
041002  037770                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

041004  036166                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

041006  036116                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
041010  036440                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
041012  036560                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
041014  040516                          $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
041016  040554                          $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;**********************************************************************
                                ;POWER DOWN ROUTINE
041020  012737  041176  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
041026  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
041034  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
041036  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
041040  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
041042  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
041044  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
```

```
                                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
                                        MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
                                        MOV     SP,$SAVR6       ;;SAVE SP
                                        MOV     #$PWRUP,@#PWRVEC;;SET UP VECTOR
                                        HALT
                                        BR      .-2             ;;HANG UP

                                ;;**************************************************************
                                ;POWER UP ROUTINE
                                $PWRUP: MOV     #$ILLUP,@#PWRVEC;;SET FOR FAST DOWN
                                        MOV     $SAVR6,SP       ;;GET SP
                                        CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
                                1$:     INC     $SAVR6          ;;WAIT FOR THE INC
                                        BNE     1$              ;;OF  WORD
                                        MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
                                        MOV     (SP)+,R5        ;;POP STACK INTO R5
                                        MOV     (SP)+,R4        ;;POP STACK INTO R4
                                        MOV     (SP)+,R3        ;;POP STACK INTO R3
                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                                        MOV     #$PWRDN,@#PWRVEC;;SET UP THE POWER DOWN VECTOR
                                        MOV     #340,@#PWRVEC+2 ;;PRIO:7
                                        TYPE                    ;;REPORT THE POWER FAILURE
                                $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
                                        MOV     (PC)+,(SP)      ;;RESTART AT START
                                $PWRAD: .WORD   START           ;;RESTART ADDRESS
                                        BIC     #20,2(SP)       ;;CLEAR "T" BIT
                                        CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
                                        RTI
                                $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
                                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
                                $SAVR6: 0                       ;;PUT THE SP HERE
                                PWRMSG: .ASCIZ  <12><15>? POWER FAILURE - RESTARTING ?<12><15>

                                        .EVEN
```

```
                                        .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
041246  047125  054105  042520  EM1:    .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
041254  052103  042105  041440
041262  052520  052040  040522
041270  020120  047524  046040
041276  041517  020056  030060
041304  000064
041306  047125  054105  042520  EM2:    .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
041314  052103  042105  046440
041322  046505  020056  043515
041330  052115  020056  051124
041336  050101  052040  020117
041344  047514  027103  031040
041352  030065      000
041355      120 044522  051117  EM3:    .ASCIZ  /PRIORITY BITS SET WRONG IN PSW/
041362  052111  020131  044502
041370  051524  051440  052105
041376  053440  047522  043516
041404  044440  020116  051520
041412  000127
041414  047515  042504  041040  EM4:    .ASCIZ  /MODE BITS SET WRONG IN PSW/
041422  052111  020123  042523
041430  020124  051127  047117
041436  020107  047111  050040
041444  053523      000
041447      104 040525  020114  EM5:    .ASCIZ  /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
041454  042101  051104  051505
041462  044523  043516  041040
041470  052105  042527  047105
041476  044040  023111  047514
041504  041040  052131  051505
041512  047440  020106  051520
041520  000127
041522  042513  047122  046105  EM6:    .ASCIZ  /KERNEL R6 CHANGED BY WRITING USER R6/
041530  051040  020066  044103
041536  047101  042507  020104
041544  054502  053440  044522
041552  044524  043516  052440
041560  042523  020122  033122
041566      000
041567      101 046440  046505  EM7:    .ASCIZ  /A MEMORY MGMT. REG. TIMED OUT/
041574  051117  020131  043515
041602  052115  020056  042522
041610  027107  052040  046511
041616  042105  047440  052125
041624      000
041625      123 046525  040515  EM10:   .ASCIZ  /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
041632  054522  047440  020106
041640  042515  027115  046440
041646  046507  027124  051040
041654  043505  020056  044524
041662  042515  052517  051524
041670      000
041671      113 046505  020056  EM11:   .ASCIZ  /MEM. MGMT. REG. WOULD NOT CLEAR/
041676  043515  052115  020056
041704  042522  027107  053440
```

```
5379  041712  052517  042114  047040
5380  041720  052117  041440  042514
5381  041726  051101     000
5382  041731     115  046505  020056  EM12:   .ASCIZ  /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
5383  041736  043515  052115  020056
5384  041744  042522  027107  041040
5385  041752  052111  020123  047516
5386  041760  020124  042523  020124
5387  041766  047503  051122  041505
5388  041774  046124  000131
5389  042000  051123  020060  043105  EM13:   .ASCIZ  /SR0 EFFECTED BY WRITE TO PSW/
5390  042006  042506  052103  042105
5391  042014  041040  020131  051127
5392  042022  052111  020105  047524
5393  042030  050040  053523     000
5394  042035     123  030522  042040  EM14:   .ASCIZ  /SR1 DID NOT READ ALL ZEROS/
5395  042042  042111  047040  052117
5396  042050  051040  040505  020104
5397  042056  046101  020114  042532
5398  042064  047522  000123
5399  042070  052504  046101  040440  EM15:   .ASCIZ  /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5400  042076  042104  042522  051523
5401  042104  047111  020107  042502
5402  042112  053524  042505  020116
5403  042120  054502  042524  020123
5404  042126  043117  050040  051101
5405  042134  047440  020122  042120
5406  042142  000122
5407  042144  052504  046101  040440  EM16:   .ASCIZ  /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5408  042152  042104  042522  051523
5409  042160  047111  020107  042502
5410  042166  053524  042505  020116
5411  042174  040520  026522  042120
5412  042202  023522  000123
5413  042206  044120  051531  020056  EM17:   .ASCIZ  /PHYS. ADDR. FORMED WRONG IN MAINT. MODE/
5414  042214  042101  051104  020056
5415  042222  047506  046522  042105
5416  042230  053440  047522  043516
5417  042236  044440  020116  040515
5418  042244  047111  027124  046440
5419  042252  042117  000105
5420  042256  044120  051531  020056  EM20:   .ASCIZ  /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
5421  042264  042101  051104  020056
5422  042272  047506  046522  042105
5423  042300  053440  047522  043516
5424  042306  044440  020116  042522
5425  042314  047514  040503  042524
5426  042322  046440  042117  000105
5427  042330  026527  044502  020124  EM21:   .ASCIZ  /W-BIT DID NOT GET SET IN PDR/
5428  042336  044504  020104  047516
5429  042344  020124  042507  020124
5430  042352  042523  020124  047111
5431  042360  050040  051104     000
5432  042365     127  041055  052111  EM22:   .ASCIZ  /W-BIT SET IN MORE THAN ONE PDR/
5433  042372  051440  052105  044440
5434  042400  020116  047515  042522
```

```
5435  042406  052040  040510  020116
5436  042414  047117  020105  042120
5437  042422  000122
5438  042424  026527  044502  020124  EM23:   .ASCIZ  /W-BIT NOT CLEARED BY WRITING TO PDR/
5439  042432  047516  020124  046103
5440  042440  040505  042522  020104
5441  042446  054502  053440  044522
5442  042454  044524  043516  052040
5443  042462  020117  042120  000122
5444  042470  051127  052111  047111  EM24:   .ASCIZ  /WRITING SR0 SET W-BIT IN KIPDR7/
5445  042476  020107  051123  020060
5446  042504  042523  020124  026527
5447  042512  044502  020124  047111
5448  042520  045440  050111  051104
5449  042526  000067
5450  042530  026527  044502  020124  EM25:   .ASCIZ  /W-BIT GOT SET DURING ODD ADDR. ABORT/
5451  042536  047507  020124  042523
5452  042544  020124  052504  044522
5453  042552  043516  047440  042104
5454  042560  040440  042104  027122
5455  042566  040440  047502  052122
5456  042574     000
5457  042575     115  046505  051117  EM26:   .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
5458  042602  020131  043515  052115
5459  042610  020056  041501  042503
5460  042616  051523  040440  047502
5461  042624  052122  042040  042111
5462  042632  047040  052117  047440
5463  042640  041503  051125     000
5464  042645     101  041503  051505  EM27:   .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5465  042652  020123  051105  047522
5466  042660  020122  044504  020104
5467  042666  047516  020124  041101
5468  042674  051117  020124  047111
5469  042702  052123  052522  052103
5470  042710  047511  000116
5471  042714  051123  020060  044504  EM30:   .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
5472  042722  020104  047516  020124
5473  042730  042522  047520  052122
5474  042736  040440  041503  051505
5475  042744  020123  051105  047522
5476  042752  020122  047503  051122
5477  042760  041505  046124  000131
5478  042766  044504  020104  047516  EM31:   .ASCIZ  /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
5479  042774  020124  047514  045503
5480  043002  050125  041440  051117
5481  043010  042522  052103  053040
5482  043016  051111  052524  046101
5483  043024  040440  042104  027122
5484  043032     000
5485  043033     120  043501  020105  EM32:   .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5486  043040  043514  044124  020056
5487  043046  041101  051117  020124
5488  043054  041517  052503  051122
5489  043062  042105  053440  042510
5490  043070  020116  052111  051440
```

```
5491  043076  047510  046125  047104
5492  043104  052047  044040  053101
5493  043112  000105
5494  043114  040520  042507  046040  EM33:   .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5495  043122  052107  027110  040440
5496  043130  047502  052122  042040
5497  043136  042111  047040  052117
5498  043144  047440  041503  051125
5499  043152  053440  042510  020116
5500  043160  052111  051440  047510
5501  043166  046125  020104  040510
5502  043174  042526     000
5503  043177     123  030122  042040  EM34:   .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
5504  043204  042111  047040  052117
5505  043212  051040  050105  051117
5506  043220  020124  040520  042507
5507  043226  046040  052107  027110
5508  043234  040440  047502  052122
5509  043242  041440  051117  042522
5510  043250  052103  054514     000
5511  043255     123  030122  047440  EM37:   .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
5512  043262  020122  051123  020062
5513  043270  044103  047101  042507
5514  043276  020104  054502  040440
5515  043304  051440  041505  047117
5516  043312  020104  041101  051117
5517  043320  000124
5518  043322  051123  020060  051117  EM40:   .ASCIZ  /SR0 OR SR2 WERE NOT "RESET" BY A RESET/
5519  043330  051440  031122  053440
5520  043336  051105  020105  047516
5521  043344  020124  051042  051505
5522  043352  052105  020042  054502
5523  043360  040440  051040  051505
5524  043366  052105     000
5525  043371     123  031122  047040  EM41:   .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
5526  043376  052117  052040  040522
5527  043404  045503  047111  020107
5528  043412  047503  051122  041505
5529  043420  046124  000131
5530  043424  044504  020104  047516  EM42:   .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
5531  043432  020124  051124  050101
5532  043440  052040  051110  020125
5533  043446  042513  047122  046105
5534  043454  051440  040520  042503
5535  043462     000
5536  043463     113  020124  051105  EM43:   .ASCIZ  /KT ERROR SERVICED ON ODD ADDR. ERROR/
5537  043470  047522  020122  042523
5538  043476  053122  041511  042105
5539  043504  047440  020116  042117
5540  043512  020104  042101  051104
5541  043520  020056  051105  047522
5542  043526  000122
5543  043530  051123  020060  051117  EM44:   .ASCIZ  /SR0 OR SR2 CHANGED BY ODD ADDR. ERROR/
5544  043536  051440  031122  041440
5545  043544  040510  043516  042105
5546  043552  041040  020131  042117
```

```
5547  043560  020104  042101  051104
5548  043566  020056  051105  047522
5549  043574  000122
5550  043576  051105  047522  020122  EM45:   .ASCIZ  /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)/
5551  043604  052504  044522  043516
5552  043612  021040  047504  041125
5553  043620  042514  042440  051122
5554  043626  051117  020042  045450
5555  043634  020124  020046  042117
5556  043642  020104  042101  051104
5557  043650  024456     000
5558  043653     115  050106  020111  EM46:   .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
5559  043660  047111  052123  052522
5560  043666  052103  047511  020116
5561  043674  052520  044123  042105
5562  043702  053440  047522  043516
5563  043710  042040  052101  000101
5564  043716  052115  044520  044440  EM47:   .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
5565  043724  051516  051124  041525
5566  043732  044524  047117  046040
5567  043740  040517  042504  020104
5568  043746  051127  047117  020107
5569  043754  040504  040524     000
5570  043761     123  040524  045503  EM50:   .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
5571  043766  047040  052117  050040
5572  043774  051525  042510  020104
5573  044002  054502  046440  050106
5574  044010  026511  052115  044520
5575  044016     000
5576  044017     113  051105  042516  EM51:   .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
5577  044024  020114  040520  042507
5578  044032  040440  041503  051505
5579  044040  020123  047111  052123
5580  044046  040505  020104  043117
5581  044054  052440  042523  035122
5582  044062  046440  050106  026511
5583  044070  052115  044520     000
5584  044075     127  047522  043516  EM52:   .ASCIZ  /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5585  044102  050040  051104  051447
5586  044110  051040  043105  051105
5587  044116  047105  042503  020104
5588  044124  044127  046111  020105
5589  044132  047111  051040  046105
5590  044140  041517  052101  020105
5591  044146  047515  042504     000
5592  044153     115  050106  020104  EM53:   .ASCIZ  /MFPD INSTRUCTION PUSHED WRONG DATA/
5593  044160  047111  052123  052522
5594  044166  052103  047511  020116
5595  044174  052520  044123  042105
5596  044202  053440  047522  043516
5597  044210  042040  052101  000101
5598  044216  052123  041501  020113  EM54:   .ASCIZ  /STACK NOT PUSHED BY MFPD-MTPD/
5599  044224  047516  020124  052520
5600  044232  044123  042105  041040
5601  044240  020131  043115  042120
5602  044246  046455  050124  000104
```

```
5603  044254  040520  020122  051117  EM55:   .ASCIZ  /PAR OR PDR CHANGED BY A RESET/
5604  044262  050040  051104  041440
5605  044270  040510  043516  042105
5606  044276  041040  020131  020101
5607  044304  042522  042523  000124
5608  044312  046111  042514  040507  EM56:   .ASCIZ  /ILLEGAL MODE 01 NOT ABORTED/
5609  044320  020114  047515  042504
5610  044326  030040  020061  047516
5611  044334  020124  041101  051117
5612  044342  042524  000104
5613  044346  051123  020060  044504  EM57:   .ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 01 CORRECTLY/
5614  044354  020104  047516  020124
5615  044362  042522  047520  052122
5616  044370  044440  046114  043505
5617  044376  046101  046440  042117
5618  044404  020105  030460  041440
5619  044412  051117  042522  052103
5620  044420  054514     000
5621  044423     120  053523  041440  EM60:   .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
5622  044430  040510  043516  042105
5623  044436  041040  020131  047101
5624  044444  051040  044524  044440
5625  044452  020116  051525  051105
5626  044460  046440  042117  000105
5627  044466  040515  047111  027124  EM61:   .ASCIZ  /MAINT. MODE (SR0 <8>) NOT DISABLED BY A RESET/
5628  044474  046440  042117  020105
5629  044502  051450  030122  036040
5630  044510  037070  020051  047516
5631  044516  020124  044504  040523
5632  044524  046102  042105  041040
5633  044532  020131  020101  042522
5634  044540  042523  000124
5635  044544  040504  040524  044440  EM62:   .ASCIZ  /DATA INCORRECT AFTER A MAINT. MODE WRITE/
5636  044552  041516  051117  042522
5637  044560  052103  040440  052106
5638  044566  051105  040440  046440
5639  044574  044501  052116  020056
5640  044602  047515  042504  053440
5641  044610  044522  042524     000
5642  044615     123  052517  041522  EM63:   .ASCIZ  /SOURCE RELOCATED IN MAINT. MODE/
5643  044622  020105  042522  047514
5644  044630  040503  042524  020104
5645  044636  047111  046440  044501
5646  044644  052116  020056  047515
5647  044652  042504     000
5648
5649  044655     117  042114  050040  DH1:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC/
5650  044662  020103  047440  042114
5651  044670  050040  053523  051040
5652  044676  020066  040527  020123
5653  044704  052040  051505  047124
5654  044712  020117  042440  051122
5655  044720  051117  041520     000
5656  044725     117  042114  050040  DH2:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC/
5657  044732  020103  047440  042114
5658  044740  050040  053523  051040
```

```
5659  044746  020066  040527  020123
5660  044754  051440  030122  020040
5661  044762  020040  051440  031122
5662  044770  020040  020040  052040
5663  044776  051505  047124  020117
5664  045004  042440  051122  051117
5665  045012  041520     000
5666  045015     127  047522  042524  DH3:    .ASCIZ  /WROTE   READ    TESTNO  ERRORPC/
5667  045022  020040  051040  040505
5668  045030  020104  020040  052040
5669  045036  051505  047124  020117
5670  045044  042440  051122  051117
5671  045052  041520     000
5672  045055     101  042104  042522  DH7:    .ASCIZ  /ADDRESS TESTNO  ERRORPC/
5673  045062  051523  052040  051505
5674  045070  047124  020117  042440
5675  045076  051122  051117  041520
5676  045104     000
5677  045105     122  043505  051511  DH10:   .ASCII  /REGISTER-ADDRS  NUM  OF/<CRLF>
5678  045112  042524  026522  042101
5679  045120  051104  020123  047040
5680  045126  046525  020040  043117
5681  045134     200
5682  045135     101  042116  042455          .ASCIZ  /AND-ED  OR-ED   TIMOUTS TESTNO  ERRORPC/
5683  045142  020104  047440  026522
5684  045150  042105  020040  052040
5685  045156  046511  052517  051524
5686  045164  052040  051505  047124
5687  045172  020117  042440  051122
5688  045200  051117  041520     000
5689  045205     122  043505  051511  DH11:   .ASCII  /REGISTR READ    READ-(BINARY)/<CRLF>
5690  045212  051124  051040  040505
5691  045220  020104  020040  051040
5692  045226  040505  026504  041050
5693  045234  047111  051101  024531
5694  045242     200
5695  045243     101  042104  042522          .ASCIZ  /ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC/
5696  045250  051523  024040  041517
5697  045256  040524  024514  032440
5698  045264  031464  030462  034460
5699  045272  033470  032466  031464
5700  045300  030462  020060  052040
5701  045306  051505  047124  020117
5702  045314  042440  051122  051117
5703  045322  041520     000
5704  045325     122  043505  051511  DH12:   .ASCII  /REGISTR WROTE   READ    READ-(BINARY)/<CRLF>
5705  045332  051124  053440  047522
5706  045340  042524  020040  051040
5707  045346  040505  020104  020040
5708  045354  051040  040505  026504
5709  045362  041050  047111  051101
5710  045370  024531     200
5711  045373     101  042104  042522          .ASCIZ  /ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC/
5712  045400  051523  024040  041517
5713  045406  040524  024514  024040
5714  045414  041517  040524  024514
```

```
5715  045422  032440  031464  030462
5716  045430  034460  033470  032466
5717  045436  031464  030462  020060
5718  045444  052040  051505  047124
5719  045452  020117  042440  051122
5720  045460  051117  041520     000
5721  045465     122  040505  020104  DH13:   .ASCIZ  /READ    TESTNO  ERRORPC/
5722  045472  020040  052040  051505
5723  045500  047124  020117  042440
5724  045506  051122  051117  041520
5725  045514     000
5726  045515     120  051101  050055  DH16:   .ASCII  /PAR-PDR PAR-PDR/<CRLF>
5727  045522  051104  050040  051101
5728  045530  050055  051104     200
5729  045535     103  042514  051101          .ASCIZ  /CLEARED EFFECTD EXPECTD RECEIVD TESTNO  ERRORPC/
5730  045542  042105  042440  043106
5731  045550  041505  042124  042440
5732  045556  050130  041505  042124
5733  045564  051040  041505  044505
5734  045572  042126  052040  051505
5735  045600  047124  020117  042440
5736  045606  051122  051117  041520
5737  045614     000
5738  045615     120  054510  044523  DH17:   .ASCII  /PHYSICL VIRTUAL/<CRLF>
5739  045622  046103  053040  051111
5740  045630  052524  046101     200
5741  045635     101  042104  042522          .ASCIZ  /ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC/
5742  045642  051523  040440  042104
5743  045650  042522  051523  045440
5744  045656  050111  051101  020064
5745  045664  052040  051505  047124
5746  045672  020117  042440  051122
5747  045700  051117  041520     000
5748  045705     120  054510  044523  DH20:   .ASCII  /PHYSICL PAR 4   PAR 5/<CRLF>
5749  045712  046103  050040  051101
5750  045720  032040  020040  050040
5751  045726  051101  032440     200
5752  045733     101  042104  042522          .ASCIZ  /ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO  ERRORPC/
5753  045740  051523  053040  040502
5754  045746  020040  020040  053040
5755  045754  040502  020040  020040
5756  045762  050040  051101  032040
5757  045770  020040  050040  051101
5758  045776  032440  020040  050040
5759  046004  053523  020040  020040
5760  046012  052040  051505  047124
5761  046020  020117  042440  051122
5762  046026  051117  041520     000
5763  046033     120  051104  020040  DH21:   .ASCII  /PDR     VIRTUAL/<CRLF>
5764  046040  020040  053040  051111
5765  046046  052524  046101     200
5766  046053     124  051505  042524          .ASCIZ  /TESTED  ADDRESS TESTNO  ERRORPC/
5767  046060  020104  040440  042104
5768  046066  042522  051523  052040
5769  046074  051505  047124  020117
5770  046102  042440  051122  051117
```

```
5771  046110  041520     000
5772  046113     120  051104  044440  DH22:   .ASCII  /PDR IN  PDR     VIRTUAL/
5773  046120  020116  050040  051104
5774  046126  020040  020040  053040
5775  046134  051111  052524  046101
5776  046142  051105  047522  020122          .ASCIZ  /ERROR   TESTED  ADDRESS TESTNO  ERRORPC/
5777  046150  020040  042524  052123
5778  046156  042105  020040  042101
5779  046164  051104  051505  020123
5780  046172  042524  052123  047516
5781  046200  020040  051105  047522
5782  046206  050122  000103
5783  046212  042120  020122  020040  DH23:   .ASCIZ  /PDR     TESTNO  ERRORPC/
5784  046220  020040  042524  052123
5785  046226  047516  020040  051105
5786  046234  047522  050122  000103
5787  046242  042120  020122  040527  DH24:   .ASCIZ  /PDR WAS EXPECTD TESTNO  ERRORPC/
5788  046250  020123  054105  042520
5789  046256  052103  020104  042524
5790  046264  052123  047516  020040
5791  046272  051105  047522  050122
5792  046300  000103
5793  046302  042120  020122  020064  DH26:   .ASCIZ  /PDR 4   PSW     TESTNO  ERRORPC/
5794  046310  020040  051520  020127
5795  046316  020040  020040  042524
5796  046324  052123  047516  020040
5797  046332  051105  047522  050122
5798  046340  000103
5799  046342  051123  020060  040527  DH30:   .ASCIZ  /SR0 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
5800  046350  020123  054105  042520
5801  046356  052103  020104  042120
5802  046364  020122  020064  020040
5803  046372  051520  020127  020040
5804  046400  020040  042524  052123
5805  046406  047516  020040  051105
5806  046414  047522  050122  000103
5807  046422  051123  020062  040527  DH31:   .ASCIZ  /SR2 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
5808  046430  020123  054105  042520
5809  046436  052103  020104  042120
5810  046444  020122  020064  020040
5811  046452  051520  020127  020040
5812  046460  020040  042524  052123
5813  046466  047516  020040  051105
5814  046474  047522  050122  000103
5815  046502  027126  027102  027101  DH32:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC/
5816  046510  020040  044513  042120
5817  046516  032122  020040  051123
5818  046524  020060  040527  020123
5819  046532  051123  020062  040527
5820  046540  020123  042524  052123
5821  046546  047516  020040  051105
5822  046554  047522  050122  000103
5823  046562  027126  027102  027101  DH33:   .ASCIZ  /V.B.A.  KIPDR4  TESTNO  ERRORPC/
5824  046570  020040  044513  042120
5825  046576  032122  020040  042524
5826  046604  052123  047516  020040
```

```
5827  046612  051105  047522  050122
5828  046620  000103
5829  046622  027126  027102  027101  DH34:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC/
5830  046630  020040  044513  042120
5831  046636  032122  020040  051123
5832  046644  020060  040527  020123
5833  046652  054105  042520  052103
5834  046660  020104  042524  052123
5835  046666  047516  020040  051105
5836  046674  047522  050122  000103
5837  046702  027126  027102  027101  DH35:   .ASCIZ  /V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC/
5838  046710  020040  044513  042120
5839  046716  032122  020040  051123
5840  046724  020062  040527  020123
5841  046732  054105  042520  052103
5842  046740  020104  042524  052123
5843  046746  047516  020040  051105
5844  046754  047522  050122  000103
5845  046762  051123  020062  040527  DH36:   .ASCIZ  /SR2 WAS EXPECTD TESTNO  ERRORPC/
5846  046770  020123  054105  042520
5847  046776  052103  020104  042524
5848  047004  052123  047516  020040
5849  047012  051105  047522  050122
5850  047020  000103
5851  047022  044506  051522  020124  DH37:   .ASCII  /FIRST ABORT     SECOND ABORT/<CRLF>
5852  047030  041101  051117  020124
5853  047036  020040  020040  042523
5854  047044  047503  042116  040440
5855  047052  047502  052122     200
5856  047057     123  030122  053440          .ASCIZ  /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC/
5857  047064  051501  051440  031122
5858  047072  053440  051501  051440
5859  047100  030122  053440  051501
5860  047106  051440  031122  053440
5861  047114  051501  052040  051505
5862  047122  047124  020117  042440
5863  047130  051122  051117  041520
5864  047136     000
5865  047137     123  030122  053440  DH40:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
5866  047144  051501  051440  031122
5867  047152  053440  051501  052040
5868  047160  051505  047124  020117
5869  047166  042440  051122  051117
5870  047174  041520     000
5871  047177     120  053523  053440  DH42:   .ASCIZ  /PSW WAS R6 WAS  TESTNO  ERRORPC/
5872  047204  051501  051040  020066
5873  047212  040527  020123  052040
5874  047220  051505  047124  020117
5875  047226  042440  051122  051117
5876  047234  041520     000
5877  047237     105  050130  041505  DH44:   .ASCII  /EXPECTED        RECEIVED/<CRLF>
5878  047244  042524  020104  020040
5879  047252  020040  020040  020040
5880  047260  042522  042503  053111
5881  047266  042105     200
5882  047271     123  030122  020040          .ASCIZ  /SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC/
```

```
5883   047276  020040  051440  031122
5884   047304  020040  020040  051440
5885   047312  030122  053440  051501
5886   047320  051440  031122  053440
5887   047326  051501  052040  051505
5888   047334  047124  020117  042440
5889   047342  051122  051117  041520
5890   047350     000
5891   047351     105  050130  041505  DH45:   .ASCII  /EXPECTED:/<CRLF>
5892   047356  042524  035104     200
5893   047363     120  053523  020040          .ASCII  /PSW     PC      SR0     SR2/<CRLF>
5894   047370  020040  050040  020103
5895   047376  020040  020040  051440
5896   047404  030122  020040  020040
5897   047412  051440  031122     200
5898   047417     061  030067  030460          .ASCII  /170017  (3$+4)  020147  (3$)/<CRLF>
5899   047424  020067  024040  022063
5900   047432  032053  020051  030040
5901   047440  030062  032061  020067
5902   047446  024040  022063  100051
5903   047454  042522  042503  053111          .ASCII  /RECEIVED:/<CRLF>
5904   047462  042105  100072
5905   047466  051520  020127  020040          .ASCIZ  /PSW     PC      SR0     SR2     TESTNO  ERRORPC/
5906   047474  020040  041520  020040
5907   047502  020040  020040  051123
5908   047510  020060  020040  020040
5909   047516  051123  020062  020040
5910   047524  020040  042524  052123
5911   047532  047516  020040  051105
5912   047540  047522  050122  000103
5913   047546  040504  040524  020040  DH46:   .ASCII  /DATA    DATA/<CRLF>
5914   047554  020040  040504  040524
5915   047562     200
5916   047563     105  050130  041505          .ASCIZ  /EXPECTD RECEIVD TESTNO  ERRORPC/
5917   047570  042124  051040  041505
5918   047576  044505  042126  052040
5919   047604  051505  047124  020117
5920   047612  042440  051122  051117
5921   047620  041520     000
5922   047623     124  051505  047124  DH50:   .ASCIZ  /TESTNO  ERRORPC/
5923   047630  020117  042440  051122
5924   047636  051117  041520     000
5925   047643     123  030122  053440  DH51:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
5926   047650  051501  051440  031122
5927   047656  053440  051501  052040
5928   047664  051505  047124  020117
5929   047672  042440  051122  051117
5930   047700  041520     000
5931   047703     120  054510  044523  DH52:   .ASCII  /PHYSICL PAR 4/<CRLF>
5932   047710  046103  050040  051101
5933   047716  032040     200
5934   047721     101  042104  042522          .ASCIZ  /ADDRESS V.B.A.  PAR 4   SR0 WAS SR2 WAS PSW     TESTNO  ERRORPC/
5935   047726  051523  053040  041056
5936   047734  040456  020056  050040
5937   047742  051101  032040  020040
5938   047750  051440  030122  053440
```

```
 5939  047756  051501  051440  031122
 5940  047764  053440  051501  050040
 5941  047772  053523  020040  020040
 5942  050000  052040  051505  047124
 5943  050006  020117  042440  051122
 5944  050014  051117  041520     000
 5945  050021     123  030122  053440  DH57:   .ASCIZ  /SRO WAS EXPECTD TESTNO  ERRORPC/
 5946  050026  051501  042440  050130
 5947  050034  041505  042124  052040
 5948  050042  051505  047124  020117
 5949  050050  042440  051122  051117
 5950  050056  041520     000
 5951  050061     120  053523  053440  DH60:   .ASCIZ  /PSW WAS EXPECTD TESTNO  ERRORPC/
 5952  050066  051501  042440  050130
 5953  050074  041505  042124  052040
 5954  050102  051505  047124  020117
 5955  050110  042440  051122  051117
 5956  050116  041520     000
 5957
 5958          050122                  .EVEN
 5959
 5960  050122  001356  001360  001354  DT1:    .WORD   TRAPPC,TRAPPS,WASR6,TESTNO,$ERRPC,0
 5961  050130  001352  001116  000000
 5962  050136  001356  001360  001354  DT2:    .WORD   TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5963  050144  001362  001364  001352
 5964  050152  001116  000000
 5965  050156  001162  001164  001352  DT3:    .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
 5966  050164  001116  000000
 5967  050170  001162  001352  001116  DT7:    .WORD   $REG0,TESTNO,$ERRPC,0
 5968  050176  000000
 5969  050200  001370  001372  001374  DT10:   .WORD   ANDADR,ORADR,TONUM,TESTNO,$ERRPC,0
 5970  050206  001352  001116  000000
 5971  050214  001162  001164  001164  DT11:   .WORD   $REG0,$REG1,$REG1,TESTNO,$ERRPC,0
 5972  050222  001352  001116  000000
 5973  050230  001162  001164  001166  DT12:   .WORD   $REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
 5974  050236  001166  001352  001116
 5975  050244  000000
 5976  050246  001162  001352  001116  DT13:   .WORD   $REG0,TESTNO,$ERRPC,0
 5977  050254  000000
 5978  050256  001162  001164  001174  DT16:   .WORD   $REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
 5979  050264  001166  001352  001116
 5980  050272  000000
 5981  050274  001402  001376  001172  DT17:   .WORD   PBALO,VIRT1,$REG4,TESTNO,$ERRPC,0
 5982  050302  001352  001116  000000
 5983  050310  001402  001376  001400  DT20:   .WORD   PBALO,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
 5984  050316  001172  001174  001176
 5985  050324  001352  001116  000000
 5986  050332  001174  001170  001352  DT21:   .WORD   $REG5,$REG3,TESTNO,$ERRPC,0
 5987  050340  001116  000000
 5988  050344  001162  001174  001170  DT22:   .WORD   $REG0,$REG5,$REG3,TESTNO,$ERRPC,0
 5989  050352  001352  001116  000000
 5990  050360  001174  001352  001116  DT23:   .WORD   $REG5,TESTNO,$ERRPC,0
 5991  050366  000000
 5992  050370  001166  001164  001352  DT24:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
 5993  050376  001116  000000
 5994  050402  001166  001176  001352  DT26:   .WORD   $REG2,$TMP0,TESTNO,$ERRPC,0
```

```
 5995  050410  001116  000000
 5996  050414  001362  001170  001166  DT30:   .WORD   WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
 5997  050422  001176  001352  001116
 5998  050430  000000
 5999  050432  001364  001172  001166  DT31:   .WORD   WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
 6000  050440  001176  001352  001116
 6001  050446  000000
 6002  050450  001162  001172  001362  DT32:   .WORD   $REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6003  050456  001364  001352  001116
 6004  050464  000000
 6005  050466  001162  001172  001352  DT33:   .WORD   $REG0,$REG4,TESTNO,$ERRPC,0
 6006  050474  001116  000000
 6007  050500  001162  001172  001362  DT34:   .WORD   $REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
 6008  050506  001166  001352  001116
 6009  050514  000000
 6010  050516  001162  001172  001364  DT35:   .WORD   $REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
 6011  050524  001170  001352  001116
 6012  050532  000000
 6013  050534  001364  001164  001352  DT36:   .WORD   WASSR2,$REG1,TESTNO,$ERRPC,0
 6014  050542  001116  000000
 6015  050546  001176  001202  001362  DT37:   .WORD   $TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6016  050554  001364  001352  001116
 6017  050562  000000
 6018  050564  001362  001364  001352  DT40:   .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
 6019  050572  001116  000000
 6020  050576  001164  001166  001352  DT42:   .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
 6021  050604  001116  000000
 6022  050610  001162  001164  001362  DT44:   .WORD   $REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6023  050616  001364  001352  001116
 6024  050624  000000
 6025  050626  001164  001170  001362  DT45:   .WORD   $REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6026  050634  001364  001352  001116
 6027  050642  000000
 6028  050644  001162  001164  001352  DT46:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
 6029  050652  001116  000000
 6030  050656  001352  001116  000000  DT50:   .WORD   TESTNO,$ERRPC,0
 6031  050664  001362  001364  001352  DT51:   .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
 6032  050672  001116  000000
 6033  050676  001402  001376  001172  DT52:   .WORD   PBALO,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
 6034  050704  001362  001364  001176
 6035  050712  001352  001116  000000
 6036  050720  001362  001164  001352  DT57:   .WORD   WASSR0,$REG1,TESTNO,$ERRPC,0
 6037  050726  001116  000000
 6038  050732  001164  001166  001352  DT60:   .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
 6039  050740  001116  000000
 6040  050744  001364  001172  001352  DT64:   .WORD   WASSR2,$REG4,TESTNO,$ERRPC,0
 6041  050752  001116  000000
 6042
 6043  050756      000      000      000  DF1:    .BYTE   0,0,0,0,0
 6044  050761      000      000
 6045  050763      000      000      000  DF2:    .BYTE   0,0,0,0,0,0,0
 6046  050766      000      000      000
 6047  050771      000
 6048  050772      000      000      000  DF3:    .BYTE   0,0,0,0
 6049  050775      000
 6050  050776      000      000      000  DF7:    .BYTE   0,0,0
```

```
 6051  051001  000  000  001  DF10:  .BYTE  0,0,1,0,0
 6052  051004  000  000
 6053  051006  000  000  002  DF11:  .BYTE  0,0,2,0,0
 6054  051011  000  000
 6055  051013  000  000  000  DF12:  .BYTE  0,0,0,2,0,0
 6056  051016  002  000  000
 6057  051021  000  000  000  DF13:  .BYTE  0,0,0
 6058  051024  000  000  000  DF16:  .BYTE  0,0,0,0,0,0
 6059  051027  000  000  000
 6060  051032  003  000  000  DF17:  .BYTE  3,0,0,0,0
 6061  051035  000  000
 6062  051037  003  000  000  DF20:  .BYTE  3,0,0,0,0,0,0,0
 6063  051042  000  000  000
 6064  051045  000  000
 6065  051047  000  000  000  DF21:  .BYTE  0,0,0,0
 6066  051052  000
 6067  051053  000  000  000  DF22:  .BYTE  0,0,0,0,0
 6068  051056  000  000
 6069  051060  000  000  000  DF23:  .BYTE  0,0,0
 6070  051063  000  000  000  DF24:  .BYTE  0,0,0,0
 6071  051066  000
 6072  051067  000  000  000  DF30:  .BYTE  0,0,0,0,0,0
 6073  051072  000  000  000
 6074  051075  000  000  000  DF46:  .BYTE  0,0,0,0
 6075  051100  000
 6076  051101  000  000       DF50:  .BYTE  0,0
 6077  051103  000  000  000  DF51:  .BYTE  0,0,0,0
 6078  051106  000
 6079  051107  003  000  000  DF52:  .BYTE  3,0,0,0,0,0,0,0
 6080  051112  000  000  000
 6081  051115  000  000
 6082  051117  000  000  000  DF57:  .BYTE  0,0,0,0
 6083  051122  000
 6084  051123  000  000  000  DF60:  .BYTE  0,0,0,0
 6085  051126  000
 6086
 6087          000001            .END
```

```
ABASE = 000000            819     860
ACDW1 = 000000            819     862
ACDW2 = 000000            819     863
ACPUOP= 000000            819     834
ADDW0 = 000000            819     864
ADDW1 = 000000            819     865
ADDW10= 000000            819     874
ADDW11= 000000            819     875
ADDW12= 000000            819     876
ADDW13= 000000            819     877
ADDW14= 000000            819     878
ADDW15= 000000            819     879
ADDW2 = 000000            819     866
ADDW3 = 000000            819     867
ADDW4 = 000000            819     868
ADDW5 = 000000            819     869
ADDW6 = 000000            819     870
ADDW7 = 000000            819     871
ADDW8 = 000000            819     872
ADDW9 = 000000            819     873
ADEVCT= 000000            819     825
ADEVM = 000000            819     861
AENV  = 000000            819     830
AENVM = 000000            819     831
AFATAL= 000000            819     822
AMADR1= 000000            819     847
AMADR2= 000000            819     851
AMADR3= 000000            819     854
AMADR4= 000000            819     857
AMAMS1= 000000            819     841
AMAMS2= 000000            819     849
AMAMS3= 000000            819     852
AMAMS4= 000000            819     855
AMSGAD= 000000            819     827
AMSGLG= 000000            819     828
AMSGTY= 000000            819     821
AMTYP1= 000000            819     842
AMTYP2= 000000            819     850
AMTYP3= 000000            819     853
AMTYP4= 000000            819     856
ANDADR  001370            892#   1544*   1566*   1584*   1606*   1624*   1646*   1664*   1686*   1704*   1726*   5969
APASS = 000000            819     824
APRIOR= 000000            819
APTCSU= 000040           4867    4972#
APTENV= 000001           4332    4860    4928    4970#
APTSIZ= 000200           1362    4969#
APTSPO= 000100           4862    4930    4971#
ASWREG= 000000            819     832
ATESTN= 000000            819     823
AUNIT = 000000            819     826
AUSWR = 000000            819     833
AVECT1= 000000            819     858
AVECT2= 000000            819     859
BIT0  = 000001            632#
BIT00 = 000001            622#    632
BIT01 = 000002            621#    631
```

```
BIT02 = 000004            620#    630
BIT03 = 000010            619#    629
BIT04 = 000020            618#    628
BIT05 = 000040            617#    627
BIT06 = 000100            616#    626
BIT07 = 000200            615#    625
BIT08 = 000400            614#    624   4260
BIT09 = 001000            613#    623   4268   4343
BIT1  = 000002            631#
BIT10 = 002000            612#   4320
BIT11 = 004000            611#   4275
BIT12 = 010000            610#   4215
BIT13 = 020000            609#   4327
BIT14 = 040000            608#   4246
BIT15 = 100000            607#   1763
BIT2  = 000004            630#
BIT3  = 000010            629#
BIT4  = 000020            628#    706
BIT5  = 000040            627#
BIT6  = 000100            626#    707
BIT7  = 000200            625#
BIT8  = 000400            624#   2171   2173   2224   2242   2308   2334   2360   2387   2410   2452   2478   2504
                         2530   2556
BIT9  = 001000            623#
BPTVEC= 000014            639#
CKSWR = 104410           4245   4309   4342   5265#
CMPREG  035612           2042   2051   2060   2069   4524#
CMSG    037167           4817   4828#
CNTRLC  037110           4663   4751   4815#
CR    = 000015            547#   4906   4916
CRLF  = 000200            548#   1386   4877   4916   5677   5689   5704   5726   5738   5748   5763   5851   5877
                         5891   5893   5898   5903   5913   5931
DDISP = 177570            554#    786   1350
DF1     050756            919   6043#
DF10    051001            962   6051#
DF11    051006            969   1201   6053#
DF12    051013            976    995   6055#
DF13    051021            982    988   6057#
DF16    051024           1002   6058#
DF17    051032           1009   6060#
DF2     050763            925   1237   6045#
DF20    051037           1016   6062#
DF21    051047           1023   6065#
DF22    051053           1030   6067#
DF23    051060           1036   1132   6069#
DF24    051063           1042   1048   1054   1060   1084   1101   1114   1120   1126   1243   6070#
DF3     050772            931    937    943    949   1231   6048#
DF30    051067           1066   1072   1078   1090   1095   1108   1139   1149   6072#
DF46    051075           1156   1163   1188   6074#
DF50    051101           1169   1194   1207   1225   6076#
DF51    051103           1175   6077#
DF52    051107           1182   6079#
DF57    051117           1213   6082#
DF60    051123           1219   6084#
DF7     050776            955   6050#
DH1     044655            917   5649#
```

L10


```
DH10    045105   959   5677#
DH11    045205   966   1198   5689#
DH12    045325   973    992   5704#
DH13    045465   980    986   5721#
DH16    045515   999   5726#
DH17    045615  1006   5738#
DH2     044725   923   1235   5656#
DH20    045705  1013   5748#
DH21    046033  1020   5763#
DH22    046113  1027   5772#
DH23    046212  1034   1130   5783#
DH24    046242  1040   1046   5787#
DH26    046302  1052   1058   5793#
DH3     045015   929    935    941    947    1229   5666#
DH30    046342  1064   5799#
DH31    046422  1070   5807#
DH32    046502  1076   5815#
DH33    046562  1082   5823#
DH34    046622  1088   5829#
DH35    046702  1093   5837#
DH36    046762  1099   1118   1241   5845#
DH37    047022  1105   5851#
DH40    047137  1112   5865#
DH42    047177  1124   5871#
DH44    047237  1136   5877#
DH45    047351  1143   5891#
DH46    047546  1153   1160   1185   5913#
DH50    047623  1167   1192   1205   1223   5922#
DH51    047643  1173   5925#
DH52    047703  1179   5931#
DH57    050021  1211   5945#
DH60    050061  1217   5951#
DH7     045055   953   5672#
DISPLA  001142   786#  1350#  1358#  4290#  4319#
DISPRE  000174   722#  1358
DSWR =  177570   553#   785   1349
DT1     050122   918   5960#
DT10    050200   961   5969#
DT11    050214   968   1200   5971#
DT12    050230   975    994   5973#
DT13    050246   981    987   5976#
DT16    050256  1001   5978#
DT17    050274  1008   5981#
DT2     050136   924   1236   5962#
DT20    050310  1015   5983#
DT21    050332  1022   5986#
DT22    050344  1029   5988#
DT23    050360  1035   1131   5990#
DT24    050370  1041   1047   5992#
DT26    050402  1053   1059   5994#
DT3     050156   930    936    942    948    1230   5965#
DT30    050414  1065   5996#
DT31    050432  1071   5999#
DT32    050450  1077   6002#
DT33    050466  1083   6005#
DT34    050500  1089   6007#
```

```
DT35    050516          1094    6010#
DT36    050534          1100    1119    6013#
DT37    050546          1107    6015#
DT40    050564          1113    6018#
DT42    050576          1125    6020#
DT44    050610          1138    6022#
DT45    050626          1148    6025#
DT46    050644          1155    1162    1187    6028#
DT50    050656          1168    1193    1206    1224    6030#
DT51    050664          1174    6031#
DT52    050676          1181    6033#
DT57    050720          1212    6036#
DT60    050732          1218    6038#
DT64    050744          1242    6040#
DT7     050170           954    5967#
EMTVEC= 000030           642#   1319#   1320#
EM1     041246           916    5324#
EM10    041625           958    5369#
EM11    041671           965    5376#
EM12    041731           972    5382#
EM13    042000           979    5389#
EM14    042035           985    5394#
EM15    042070           991    5399#
EM16    042144           998    5407#
EM17    042206          1005    5413#
EM2     041306           922    5330#
EM20    042256          1012    5420#
EM21    042330          1019    5427#
EM22    042365          1026    5432#
EM23    042424          1033    5438#
EM24    042470          1039    5444#
EM25    042530          1045    5450#
EM26    042575          1051    5457#
EM27    042645          1057    5464#
EM3     041355           928    5337#
EM30    042714          1063    5471#
EM31    042766          1069    1092    1098    1240    5478#
EM32    043033          1075    5485#
EM33    043114          1081    5494#
EM34    043177          1087    5503#
EM37    043255          1104    5511#
EM4     041414           934    5343#
EM40    043322          1111    5518#
EM41    043371          1117    5525#
EM42    043424          1123    5530#
EM43    043463          1129    5536#
EM44    043530          1135    5543#
EM45    043576          1142    5550#
EM46    043653          1152    5558#
EM47    043716          1159    5564#
EM5     041447           940    5348#
EM50    043761          1166    5570#
EM51    044017          1172    5576#
EM52    044075          1178    5584#
EM53    044153          1184    5592#
EM54    044216          1191    5598#
```

```
EM55    044254          1197    5603#
EM56    044312          1204    5608#
EM57    044346          1210    5613#
EM6     041522           946    5356#
EM60    044423          1216    5621#
EM61    044466          1222    5627#
EM62    044544          1228    5635#
EM63    044615          1234    5642#
EM7     041567           952    5363#
ERRTYP  035206          4329    4375#
ERRVEC= 000004           635#   1347    1348*   1359*   1390*   1391*   2854*   2864*   4251    4252*   4254*   4257*
FORMPA  036004          2317    2343    2369    2396    2419    2461    2487    2513    2539    2565    2636    2677    4600#
GNS     = ****** U       721    1385    4184    4191    5256    5257    5258    5259    5260    5261    5263    5265    5266
                        5267    5268    5269
GTSWR   = 104407        1380    4824    5263#
HT      = 000011         545#   4875    4916
IOTVEC= 000020           640#   1317*   1318*
KERSTK= 001100           708#   1501    1505    1507    2039    2048    2057    2066    2180    2856    3034    3106    3110
                        3123    3181    3185    3198    3266    3326    3331    3394    3413    3473    3534    3628    3960
                        4092
KIPAR0= 172340           695#   1582    1841    1935    2047    2100    2288    3592*   4501    4539    4600
KIPAR1= 172342           696#   3593#
KIPAR2= 172344           697#   3594#
KIPAR3= 172346           698#   2215#   2712    2892*   3253*   3595*
KIPAR4= 172350           699#   2216#   2306*   2332*   2358*   2385*   2409*   2450*   2476*   2502*   2528*   2554*   2606*
                        2656*   2663#   2893*   3254*   3462*   3596*   3616*   3768*   3946*
KIPAR5= 172352           700#   2607*   2657*   2664*   3301*
KIPAR6= 172354           701#
KIPAR7= 172356           702#   2217*   3597*
KIPDR0= 172300           684#   1622    1888    1983    2038    2089    2161    2169*   2295    2716    2720    2734    3601
                        4497    4525
KIPDR1= 172302           685#
KIPDR2= 172304           686#
KIPDR3= 172306           687#   2218*   2236*   2853*   2857    2896*   3099*   3255*
KIPDR4= 172310           688#   2219*   2654*   2661*   2903*   2970*   3104*   3179*   3256*   3281*   3302*   3372*   3464*
                        3494*   3615*   3621*   3766*   3792*   3794*   3809*   3811*   3824*   3826*   3840*   3842*   3858*
                        3860*   3875*   3877*   3894*   3896*   3945*
KIPDR5= 172312           689#   2655*   2662*   3100*   3303*   3373*
KIPDR6= 172314           690#
KIPDR7= 172316           691#   2220*   2842*   2845
LF      = 000012         546#   4910    4916
LOOP    020474          1388#   4226
MGMERR  002314          1283#   1392    2182    2235    2667    2950    3011    3053    3159    3234    3282    3374    3493
                        3567    3734    3905    4061
MGMFLG  002316          1284#   1298*   1396*
MMVEC   = 000250         651#   1392*   1393*   2165*   2166*   2182*   2183*   2221*   2235*   2626*   2667*   2902*   2950*
                        2969*   3011*   3026*   3053*   3103*   3116*   3159*   3178*   3191*   3234*   3259*   3282*   3323*
                        3329*   3374*   3620*   3734*   3770*   3905*   3953*   4061*
ORADR   001372           893#   1545*   1564*   1585*   1604*   1625*   1644*   1665*   1684*   1705*   1724*   5969
PBAHI   001404           898#   4619*
PBALO   001402           897#   4610*   4611*   4616*   5981    5983    6033
PIRQ    = 177772         552#
PIRQVE= 000240           646#
PR0     = 000000         569#
PR1     = 000040         570#
PR2     = 000100         571#
```

```
PR3     = 000140            572#
PR4     = 000200            573#
PR5     = 000240            574#
PR6     = 000300            575#
PR7     = 000340            576#
PS      = 177776            549#  550
PSW     = 177776            550#  1436*  1437*  1438   1442*  1450*  1462*  1464*  1465   1470*  1476*  1478*  1479
                           1483*  1500*  1502*  1504*  1800*  1810*  2381   2405   2603*  2627   2648   2658*  2660*
                           2776*  2824*  2907   2940*  2946*  2948*  2974   3007*  3009*  3027*  3393*  3398*  3405
                           3407*  3412*  3415*  3417*  3433*  3437   3439*  3524*  3533*  3535*  3537*  3625*  3646*
                           3658*  3670*  3681*  3695*  3709*  3722*  3773*  3784*  3805*  3821*  3836*  3853*  3871*
                           3888*  3944*  3955*  3974*  3987*  3999*  4010*  4023*  4037*  4050*  4062*  4089*  4132*
                           4459   4461   4601
PWRMSG  041204              5305   5314#
PWRVEC= 000024               641#  1323*  1324*  5274*  5275*  5284*  5290*  5302*  5303*
RDCHR   104411              4744   5266#
RDLIN   104412              5267#
RESREG= 104414              5212   5269#
RESVEC= 000010               636#  1334*  1341*
SAVREG= 104413              5197   5268#
SETREG  035520              2040   2049   2058   2067   2087   4496#
SR0     177572               655#  1294   1296*  1398*  1542   1744   1801*  1803   1809*  2171*  2173   2175*  2181*
                           2224*  2242*  2243   2248*  2308*  2334*  2360*  2387*  2410*  2452*  2478*  2504*  2530*
                           2556*  2624*  2678   2680*  2843   2844*  2921   2936*  2988   3003*  3036   3052*  3124
                           3140*  3147   3149*  3199   3215*  3222   3224*  3274*  3327   3332   3348   3362*  3475
                           3491*  3530   3532*  3741   3743*  3911   3913*  4069   4071*
SR1     177574               656#  1822
SR2     177576               657#  1295   1754   1775   2244   2679   2922   2989   3045   3125   3148   3200   3223
                           3267   3305   3314   3328   3333   3352   3363   3476   3531   3742   3912   4070
SR3     172516               658#
STACK   001100               540#   708    709   1315   1389   4133   4152
START   020000               725   1308#  5307
STKLMT= 177774               551#
SWR     001140               785#  1313   1349*  1351   1357*  1364*  1378   4215   4246   4260   4262   4268   4275
                           4320   4327   4339   4343   4632   4677*  5282   5295*
SWREG   000176               723#  1357   1378   4632   4645
SW0     000001               604#
SW00    000001               594#   604
SW01    000002               593#   603
SW02    000004               592#   602
SW03    000010               591#   601
SW04    000020               590#   600
SW05    000040               589#   599
SW06    000100               588#   598
SW07    000200               587#   597
SW08    000400               586#   596
SW09    001000               585#   595
SW1     000002               603#
SW10    002000               584#
SW11    004000               583#
SW12    010000               582#
SW13    020000               581#
SW14    040000               580#
SW15    100000               579#
SW2     000004               602#
SW3     000010               601#
```

```
SW4    = 000020            600#
SW5    = 000040            599#
SW6    = 000100            598#
SW7    = 000200            597#
SW8    = 000400            596#
SW9    = 001000            595#
TBIT   = 000020            706#  4459   4464   4477
TBITPS  001366             891#  1397*  4462*  4477   4479   4480*
TBITVE= 000014             637#  1331*  1332*
TESTNO  001352             885#  4316*  5960   5962   5965   5967   5969   5971   5973   5976   5978   5981   5983
                          5986   5988   5990   5992   5994   5996   5999   6002   6005   6007   6010   6013   6015
                          6018   6020   6022   6025   6028   6030   6031   6033   6036   6038   6040
TIMERR  002246            1256#  1390   1555   1595   1635   1675   1715   2864   3418   3492   3565
TIMFLG  002250            1257#  1268*  1395*
TKVEC  = 000060            644#
TOFF    035432            2160   2213   2710   2777   2839   3391   3514   4459#
TON     035466            2185   2238   2756   2823   2866   3420   3570   4477#
TONUM   001374             894#  1546*  1552   1567*  1586*  1592   1607*  1626*  1632   1647*  1666*  1672   1687*
                          1706*  1712   1727*  5969
TPVEC  = 000064            645#
TRAPPC  001356             887#  1264*  1270   1291*  1300   2246*  2254   2671*  2687   3145*  3155   3220*  3230
                          3739*  3749   3909*  3919   4067*  4077   5960   5962
TRAPPS  001360             888#  1265*  1269   1292*  1299   2247*  2253   2672*  2686   3146*  3154   3221*  3229
                          3740*  3748   3910*  3918   4068*  4076   5960   5962
TRAPVE= 000034             643#  1321*  1322*
TRTVEC= 000014             638#
TST1    020556            1409#
TST10   021522            1636   1658#
TST11   021654            1676   1698#
TST12   022006            1716   1742#
TST13   022150            1798#
TST14   022206            1820#
TST15   022224            1823   1839#
TST16   022346            1886#
TST17   022470            1933#
TST2    020630            1433#
TST20   022620            1981#
TST21   022750            2034#
TST22   023170            2084#
TST23   023340            2159#
TST24   023474            2212#
TST25   023732            2239   2286#
TST26   024462            2441#
TST27   025146            2601#
TST3    020714            1460#
TST30   025612            2669   2708#
TST31   026006            2775#
TST32   026214            2837#
TST33   026344            2889#
TST34   026640            2962#
TST35   027050            3024#
TST36   027174            3098#
TST37   027476            3175#
TST4    021036            1499#
TST40   027764            3252#
TST41   030152            3300#
```

```
TST42   030552            3390#
TST43   030740            3429#
TST44   031022            3461#
TST45   031220            3513#
TST46   031542            3591#
TST47   032410            3736    3765#
TST5    021104            1506    1538#
TST50   033270            3906    3936#
TST51   034034            4064    4088#
TST52   034140            4131#
TST6    021236            1556    1578#
TST7    021370            1596    1618#
TYPBN = 104406            4424    5261#
TYPDS = 104405            4188    4195    4417    4823    5260#
TYPE  = 104401            1372    4182    4189    4196    4322    4330    4376    4393    4395    4398    4400    4432    4437
                          4443    4643    4644    4647    4658    4668    4679    4698    4748    4757    4763    4768    4772
                          4777    4778    4780    4783    4787    4817    4821    4880    4989    5057    5132    5256#   5304
TYPOC = 104402            4384    4410    4646    4820    5257#
TYPON = 104404            5259#
TYPOS = 104403            5258#
UIPAR0= 177640             673#   1662    1871    1965    2065    2123    2608    3395*   3414*   3608*   4509    4567    4603
UIPAR1= 177642             674#   3609*
UIPAR2= 177644             675#   3610*
UIPAR3= 177646             676#   2779    2894*   3515*   3611*
UIPAR4= 177650             677#   2622*   2650*   2665*   2895*   3516*   3617*   3769*   3947*
UIPAR5= 177652             678#   2623*   2651*   2666*
UIPAR6= 177654             679#
UIPAR7= 177656             680#   1869    1963    3612*
UIPDR0= 177600             662#   1702    1918    2013    2056    2111    2615    2783    2787    2801    3605    3940    4505
                          4553
UIPDR1= 177602             663#
UIPDR2= 177604             664#
UIPDR3= 177606             665#   2897*   3517*   3568*
UIPDR4= 177610             666#   2620*   2652*   2904*   2971*   3518*   3767*   3954*
UIPDR5= 177612             667#   2621*   2653*
UIPDR6= 177614             668#
UIPDR7= 177616             669#   1916    2011
USESTK= 000700             709#   1503    3399    3416    3536    3631    3785    3957    4095    4118
VIRT1   001376             895#   2316*   2342*   2368*   2395*   2418*   2460*   2486*   2512*   2538*   2564*   2635*   2643
                          2676*   4604    4610    5981    5983    6033
VIRT2   001400             896#   2634*   5983
WASR6   001354             886#   1266*   1293*   2245*   5960    5962
WASR0   001362             889#   1294*   2243*   2678*   2921*   2923    2988*   2990    3036*   3038    3124*   3127    3147*
                          3199*   3202    3222*   3332*   3334    3348*   3349    3475*   3478    3530*   3549    3741*   3911*
                          4069*   5962    5996    6002    6007    6015    6018    6022    6025    6031    6033    6036
WASR2   001364             890#   1295*   1754*   1756    1775*   1776    2244*   2679*   2922*   2930    2989*   2997    3045*
                          3046    3125*   3134    3148*   3200*   3209    3223*   3267*   3268    3305*   3307    3314*   3316
                          3333*   3337    3352*   3353    3363*   3365    3476*   3482    3531*   3552    3742*   3912*   4070*
                          5962    5999    6002    6010    6013    6015    6018    6022    6025    6031    6033    6040
WBIT  = 000100             707#   2726    2735    2746    2793    2802    2813
$APTHD  000204             745     751#
$ASTAT= ****** U          4950    4965
$ATYC   037546            4921    4923#
$ATY1   037522            4919#
$ATY3   037530            4865    4920#
$ATY4   037540            4335    4922#
```

```
$AUTO8  001134            782#   1382*  4640   4804
$BASE   001302            860#
$BDADR  001122            777#
$BDDAT  001126            779#
$BELL   001216            811#   4322   4355
$BIN    040042           4985*   4988*  4989   4996#
$CDW1   001306            862#
$CDW2   001310            863#
$CHARC  037516           4882*   4892*  4899   4908*  4913#
$CKSWR  036116           4632#   5265
$CLR.T  034400           4201    4204#
$CMTAG  001100            765#   1310   1311   1319   1325   1326   1327
$CM1  =  000006            797#    798#   799#   800#   801#   802#   803#
$CM2  =  000014            797#    798#   799#   800#   801#   802#   803#
$CM3  =  000006            795#    797
$CM4  =  000006            803#    804#   805#   806#   807#   808#   809#
$CNTLC  037046           4658    4748   4797#
$CNTLG  037060           4643    4799#
$CNTLU  037053           4668    4772   4798#
$CPUOP  001254            834#
$CRLF   001223            813#   4196   4330   4355   4376   4395   4400   4443   4679   4777   4797   4881   4916
$DBLK   040506           5098    5132   5140#
$DB22O  040612           4429    5197#
$DDW0   001312            864#
$DDW1   001314            865#
$DDW10  001336            874#
$DDW11  001340            875#
$DDW12  001342            876#
$DDW13  001344            877#
$DDW14  001346            878#
$DDW15  001350            879#
$DDW2   001316            866#
$DDW3   001320            867#
$DDW4   001322            868#
$DDW5   001324            869#
$DDW6   001326            870#
$DDW7   001330            871#
$DDW8   001332            872#
$DDW9   001334            873#
$DEVCT  001236            825#
$DEVM   001304            861#
$DOAGN  034420           4178    4199   4206   4212#
$DTBL   040476           5101    5136#
$ENDAD  034410            732    1370   4208#  4350
$ENDCT  034236           1325    4180#
$ENULL  034464           4228#
$ENV    001246            830#   1376   4332   4860   4928   4952
$ENVM   001247            831#   1362   4862   4867   4930
$EOP    034202           4170#   4825
$EOPCT  034230           1325*   4177#  4181
$ERFLG  001103            768#   4235   4264   4266   4272*  4294   4317*  4355
$ERMAX  001115            774#   1328*  4266   4289*  4294
$ERROR  034750           1319    4308#
$ERRPC  001116            775#   4324*  4325*  4326   4355   4382   5960   5962   5965   5967   5969   5971   5973
                         5976    5978   5981   5983   5986   5988   5990   5992   5994   5996   5999   6002   6005
                         6007    6010   6013   6015   6018   6020   6022   6025   6028   6030   6031   6033   6036
```

```
                        6038    6040
$ERRTB  001406           913#    4390
$ERTTL  001112           772#    4193    4197*   4323*   4355
$ESCAP  001214           810#    1327*   4288*   4346    4348    4355
$ETABL  001246           829#
$ETEND  001352           757      882#
$FATAL  001230           822#    4956*
$FFLG   037766          4919*    4922*   4950    4959*   4967#
$FILLC  001156           793#    4885    4916
$FILLS  001155           792#    4916
$GDADR  001120           776#
$GDDAT  001124           778#
$GET42  034362          4198#
$GTSWR  036166          4644#    5263
$HD   = 000000           525
$HIBTS  000204           752#
$ICNT   001104           769#    4279*   4280    4282*   4293
$ILLUP  041176          5274     5290    5311#
$INTAG  001135           783#    4660    4680    4804
$ITEMB  001114           773#    4326*   4334    4355    4379
$LF     001224           814#    4355    4787    4797    4916
$LFLG   037765          4960*    4966#
$LOOP   034456          4220     4224#
$LPADR  001106           770#    1343*   4270*   4286*   4291    4293
$LPERR  001110           771#    1344*   1410*   1425*   1434*   1449*   1461*   1475*   1488*   1540*   1551*   1580*   1591*
                        1620*   1631*   1660*   1671*   1700*   1711*   1753*   1762*   1786*   1843*   1852*   1873*   1890*
                        1899*   1920*   1936*   1949*   1967*   1984*   1997*   2015*   2036*   2045*   2054*   2063*   2072*
                        2170*   2184*   2214*   2237*   2301*   2327*   2353*   2380*   2404*   2427*   2445*   2471*   2497*
                        2523*   2549*   2575*   2625*   2668*   2719*   2755*   2786*   2822*   2841*   2852*   2865*   2905*
                        2949*   2972*   3010*   3025*   3054*   3105*   3109*   3115*   3158*   3180*   3184*   3190*   3233*
                        3260*   3280*   3304*   3313*   3322*   3371*   3392*   3419*   3431*   3446*   3467*   3495*   3523*
                        3569*   3613*   3624*   3644*   3657*   3669*   3680*   3694*   3708*   3721*   3735*   3788*   3804*
                        3820*   3835*   3852*   3870*   3887*   3904*   3943*   3973*   3986*   3998*   4009*   4022*   4036*
                        4049*   4063*   4107*   4120*   4270    4287*   4293    4345
$MADR1  001260           847#
$MADR2  001264           851#
$MADR3  001270           854#
$MADR4  001274           857#
$MAIL   001226           753      757     820#    1361    1376    4285    4332    4860
$MAMS1  001256           841#
$MAMS2  001262           849#
$MAMS3  001266           852#
$MAMS4  001272           855#
$MBADR  000206           753#
$MFLG   037764          4920*    4926    4961*   4965#
$MNEW   037076          4647     4802#
$MSGAD  001242           827#    4936*   4939
$MSGLG  001244           828#    4941*
$MSGTY  001226           821#    4934    4942*   4954    4958*
$MSWR   037065          4644     4800#
$MTYP1  001257           842#
$MTYP2  001263           850#
$MTYP3  001267           853#
$MTYP4  001273           856#
$MXCNT  034746          4283     4293#
$NULL   001154           791#    4887    4916
```

```
$NWTST= 000001          1402#  1404   1427#  1429   1452#  1454   1490#  1492   1528#  1530   1570#  1572   1610#
                        1612   1650#  1652   1690#  1692   1730#  1732   1788#  1790   1812#  1814   1829#  1831
                        1875#  1877   1924#  1926   1969#  1971   2017#  2019   2074#  2076   2146#  2148   2199#
                        2201   2269#  2271   2429#  2431   2577#  2579   2691#  2693   2758#  2760   2826#  2828
                        2880#  2882   2953#  2955   3013#  3015   3086#  3088   3163#  3165   3238#  3240   3286#
                        3288   3378#  3380   3422#  3424   3449#  3451   3498#  3500   3579#  3581   3753#  3755
                        3924#  3926   4080#  4082   4122#  4124
$OCNT   040266          5029#  5058#  5071#
$OCTVL  040714          5199   5224#
$OMODE  040270          5024#  5028#  5033   5036#  5047#  5073#
$OVER   034732          4247   4263   4271   4281   4290#
$PASS   001234           824#  1361#  4174#  4175#  4186   4227   4277   4294   4815
$PASTM  000212           755#
$PWRAD  041160          5307#
$PWRDN  041020          1323   5274#  5302
$PWRMG  041154          5305#
$PWRUP  041072          5284   5290#
$QUES   001222           812#  4355   4698   4780   4797   4916
$RDCHR  036440          4711#  5266
$RDDEC= ****** U        5268
$RDLIN  036560          4739#  5267
$RDOCT= ****** U        5268
$ROSZ = 000010          4732#
$REGAD  001160           795#
$REGO   001162           797#  4310#  5965   5967   5971   5973   5976   5978   5988   6002   6005   6007   6010
                        6022   6028
$REG1   001164           798#  4311#  5965   5971   5973   5978   5992   6013   6020   6022   6025   6028   6036
                        6038
$REG2   001166           799#  4312#  5973   5978   5992   5994   5996   5999   6007   6020   6038
$REG3   001170           800#  4313#  5986   5988   5996   6010   6025
$REG4   001172           801#  4314#  5981   5983   5999   6002   6005   6007   6010   6033   6040
$REG5   001174           802#  4315#  5978   5983   5986   5988   5990
$RESRE  040554          5174#  5269
$RTNAD  034460          4226#
$RTRN   034454          1331   1333#  1338#  4202   4221#
$R2A  = ****** U        5270
$SAVRE  040516          5158#  5268
$SAVR6  041202          5283#  5291   5292#  5293#  5313#
$SCOPE  034470          1317   4244#
$SETUP= 000137           715#  1316   1317   1319   1321   1323   1325   1326   1327   1329   1343   1370   1373
                        4172   4245   4309   4342   4350   4627   4804   5309
$STUP = 177777           715#
$SVLAD  034676          4255   4284#
$SVPC = 000204           730#   735
$SWR  = 177400           515#   525    529    530    531    532    533    534    535    536    809    810    811
                        1326   1327   1329   1343   1344   1410   1434   1461   1500   1539   1579   1619   1659
                        1699   1743   1799   1821   1840   1887   1934   1982   2035   2085   2160   2213   2287
                        2442   2602   2709   2776   2838   2890   2963   3025   3099   3176   3253   3301   3391
                        3430   3462   3514   3592   3766   3937   4089   4132   4166   4173   4200   4213   4227
                        4236   4237   4238   4239   4240   4246   4258   4260   4261   4264   4265   4266   4273
                        4274   4275   4287   4290   4293   4300   4301   4302   4303   4304   4320   4327   4339
                        4343   4355   5308
$SWREG  001250           832#  1364
$SWRMK= 000000           536    537   4240   4241   4262
$TBIT   034462          1342#  4217#  4227#  5309#
$TESTN  001232           823#  4285#
```

```
$TIMES  001212          809#  1326*  4173*  4273*  4280   4283*  4293
$TKB    001146          788#  4625   4636   4653   4715   4721
$TKS    001144          787#  4625   4634   4650   4662*  4682*  4713   4719
$TMPO   001176          803#  2307*  2312   2333*  2338   2359*  2364   2451*  2456   2477*  2482   2503*  2508
                       2529*  2534   2555*  2560   2627*  2907*  2937   2974*  3004   3327*  3334   3538*  3544*
                       3548*  3551*  3555*  3556   5983   5994   5996   5999   6015   6033
$TMP1   001200          804#  3324*  3336*  3339*  3340   3346*  3351*  3355*  3356   3474*  3480*  3484*  3485
$TMP2   001202          805#  3328*  3337   3696*  3697   3723*  3724   3855   3856*  3890*  3892   4024*  4025
                       4051*  4052   6015
$TMP3   001204          806#
$TMP4   001206          807#
$TMP5   001210          808#  4815*  4816*  4822
$TN   = 000053          515#   525   1402   1410#  1427   1434#  1452   1461#  1490   1500#  1506   1528   1539#
                       1556   1570   1579#  1596   1610   1619#  1636   1650   1659#  1676   1690   1699#  1716
                       1730   1743#  1788   1799#  1812   1821#  1823   1829   1840#  1875   1887#  1924   1934#
                       1969   1982#  2017   2035#  2074   2085#  2146   2160#  2199   2213#  2239   2269   2287#
                       2429   2442#  2577   2602#  2669   2691   2709#  2758   2776#  2826   2838#  2880   2890#
                       2953   2963#  3013   3025#  3086   3099#  3163   3176#  3238   3253#  3286   3301#  3378
                       3391#  3422   3430#  3449   3462#  3498   3514#  3579   3592#  3736   3753   3766#  3906
                       3924   3937#  4064   4080   4089#  4122   4132#
$TPB    001152          790#  4905*  4916
$TPFLG  001157          794#  4854   4916
$TPS    001150          789#  4903   4916
$TRAP   040732         1321   5233#
$TRAP2  040754         5244#  5255
$TRP  = 000015         5248#  5257#  5258#  5259#  5260#  5261#  5262#  5263   5264#  5265   5266#  5267#  5268#
                       5269#  5270#
$TRPAD  040766         5238   5255#
$TSTM   000210          754#
$TSTNM  001102          767#  4172*  4235   4262   4284*  4285   4290   4294   4316   4319   4355   4818
$TTYIN  037036         4741   4742   4760   4778   4792   4796#
$TYPBN  037770         4982#  5261
$TYPDS  040272         5086#  5260
$TYPE   037240         4854#  4947   5248   5256
$TYPEC  037452         4684   4884   4891   4898   4903#  4904
$TYPEX  037520         4909   4911   4914#
$TYPOC  040070         5027#  5257
$TYPON  040104         5026   5029#  5259
$TYPOS  040044         5022#  5258
$UNIT   001240          826#
$UNITM  000214          756#
$USWR   001252          833#
$VECT1  001276          858#
$VECT2  001300          859#
$XTSTR  034502         4249#
$$GET4= 000001         4200#  4205
$OFILL  040267         5023#  5027*  5037   5072#
$40CAT= ****** U       4246   4329
.     = 051127          717#   721#   730    731#   733#   735#   741    742#   744#   746#   764#   815   1306#
                       1314   1343   1344   4185#  4227   4229#  4293   4294   4355   4446#  4625   4796#  4797
                       4804#  4916   4968#  5140#  5224#  5286   5312   5320#  5958#
.$ASTA= ****** U       4920   4923
.$X   = 000204          741#   746
```

```
ADDTST    714#  2301  2327  2353  2445  2471  2497  2523  2549
COMMEN    647#
ENDCOM    647#
ERROR     541#  1267  1297  1418  1443  1471  1484  1509  1554  1559  1594  1599  1634  1639  1674
         1679  1714  1719  1749  1758  1770  1779  1805  1824  1847  1860  1894  1907  1945  1957
         1993  2005  2094  2105  2116  2128  2176  2230  2249  2318  2344  2370  2397  2420  2462
         2488  2514  2540  2566  2637  2681  2728  2739  2748  2795  2806  2815  2848  2860  2909
         2917  2925  2932  2976  2984  2992  2999  3029  3040  3048  3118  3129  3136  3150  3193
         3204  3211  3225  3270  3309  3318  3342  3358  3367  3408  3442  3469  3487  3558  3634
         3639  3652  3664  3675  3687  3702  3715  3730  3744  3780  3798  3815  3830  3846  3864
         3881  3900  3914  3963  3968  3981  3993  4004  4016  4030  4043  4057  4072  4098  4103
         4114  4141  4148  4533  4547  4561  4575
ESCAPE    647#
GETPRI    647#  4213
GETSWR    647#  1373#
MSG1     1402#  1404
MSG10    1650#  1652
MSG11    1690#  1692
MSG12    1730#  1732
MSG13    1788#  1790
MSG14    1812#  1814
MSG15    1829#  1831
MSG16    1875#  1877
MSG17    1924#  1926
MSG2     1427#  1429
MSG20    1969#  1971
MSG21    2017#  2019
MSG21A   2074#  2076
MSG21B   2146#  2148
MSG21C   2199#  2201
MSG22    2269#  2271
MSG23    2429#  2431
MSG24    2577#  2579
MSG25    2691#  2693
MSG26    2758#  2760
MSG27    2826#  2828
MSG3     1452#  1454
MSG30    2880#  2882
MSG31    2953#  2955
MSG31A   3013#  3015
MSG32    3085#  3088
MSG33    3162#  3165
MSG34    3237#  3240
MSG35    3285#  3288
MSG36    3377#  3380
MSG36A   3422#  3424
MSG37    3448#  3451
MSG4     1490#  1492
MSG40    3497#  3500
MSG41    3579#  3581
MSG42    3753#  3755
MSG43    3924#  3926
MSG44    4080#  4082
MSG45    4122#  4124
MSG5     1527#  1530
MSG6     1570#  1572
```

```
MSG7      1610#  1612
MULT       647#
NEWTST     647#  1402  1427  1452  1490  1528  1570  1610  1650  1690  1730  1788  1812  1829  1875
          1924   1969  2017  2074  2146  2199  2269  2429  2577  2691  2758  2826  2880  2953  3013
          3086   3163  3238  3286  3378  3422  3449  3498  3579  3753  3924  4080  4122
POP        647#  4962  4963  5127  5179  5295  5296
PUSH       647#  4923  4925  4946  5086  5159  5276  5282
REPORT     647#
SAVR       714#  4310
SCOPE      542#  1409  1433  1460  1499  1538  1578  1618  1658  1698  1742  1798  1820  1839  1886
          1933   1981  2034  2084  2159  2212  2286  2441  2601  2708  2775  2837  2889  2962  3024
          3098   3175  3252  3300  3390  3429  3461  3513  3591  3765  3936  4088  4131  4171
SETPRI     647#
SETTRA    5248#  5257  5258  5259  5260  5261  5263  5265  5266  5267  5268  5269
SETUP      647#  1308
SKIP       647#  1506  1556  1596  1636  1676  1716  1823  2239  2669  3736  3906  4064
SLASH      647#
SPACE      647#
STARS      647#   728   738   740   747   760   815   818  1248  1255  1275  1282  1402  1408  1427
          1432   1452  1459  1490  1498  1514  1525  1528  1537  1570  1577  1610  1617  1650  1657
          1690   1697  1730  1741  1788  1797  1812  1819  1829  1838  1875  1885  1924  1932  1969
          1980   2017  2033  2074  2083  2146  2158  2199  2211  2269  2285  2429  2440  2577  2600
          2691   2707  2758  2774  2826  2836  2868  2878  2880  2888  2953  2961  3013  3023  3067
          3082   3086  3097  3163  3174  3238  3251  3286  3299  3378  3389  3422  3428  3449  3460
          3498   3512  3571  3576  3579  3590  3753  3764  3924  3935  4080  4087  4122  4130  4162
          4232   4296  4357  4451  4458  4470  4476  4486  4494  4515  4523  4584  4598  4624  4627
          4703   4732  4839  4918  4975  4999  5076  5143  5188  5227  5272  5288
SWRSU      647#  1345#
TIMMSG     714#  1572  1612  1652  1692
TIMTST     714#  1579  1619  1659  1699
TRMTRP    5248#
TYPBIN     647#  4423
TYPDEC     647#  4186  4193  4416  4822
TYPNAM     647#  1366
TYPNUM     647#
TYPOCS     647#
TYPOCT     647#  4382  4409  4645  4819
TYPTXT     647#  4182  4189
USER       714#   884
WMSG       714#  2693  2760
WTST       714#  2709  2777
SSCMRE     758#   797   798   799   800   801   802
SSCMTM     758#   803   804   805   806   807   808
SSESCA     647#
SSNEWT     647#  1402  1427  1452  1490  1528  1570  1610  1650  1690  1730  1788  1812  1829  1875
          1924   1969  2017  2074  2146  2199  2269  2429  2577  2691  2758  2826  2880  2953  3013
          3086   3163  3238  3286  3378  3422  3449  3498  3579  3753  3924  4080  4122
SSSET     5248#  5257  5258  5259  5260  5261  5263  5265  5266  5267  5268  5269
SSSETM    1361#
SSSETU    1329#
SSSKIP     647#  1506  1556  1596  1636  1676  1716  1823  2239  2669  3736  3906  4064
.EQUAT     515#   537
.HEADE     515#   516
.KT11      515#   647
.SETUP     515#   715
.SWRHI     515#   525
```

```
.SWRL0    537#
.$ACT1    515#    726
.$APTB    816#
.$APTH    515#    736
.$APTY    515#   4916
.$CATC    515#    715
.$CMTA    515#    758
.$D820    515#   5186
.$EOP     515#   4160
.$ERRO    515#   4294
.$ERRT    515#
.$POWE    515#   5270
.$READ    515#   4622
.$SAVE    515#   5141
.$SCOP    515#   4230
.$TRAP    515#   5225
.$TYPB    515#   4973
.$TYPD    515#   5074
.$TYPE    515#   4837
.$TYPO    515#   4997

. ABS.  051127     000

 ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:DFKTHA,DSKZ:DFKTHA.SEQ/CRF/SOL=DFKTHA.P11
RUN-TIME: 27 18 1 SECONDS
RUN-TIME RATIO: 310/48=6.4
CORE USED:  33K  (66 PAGES)
```