PDP11/34

MEM MGMT EXCERCISER

CFKTHBO

AH-8055B-MC

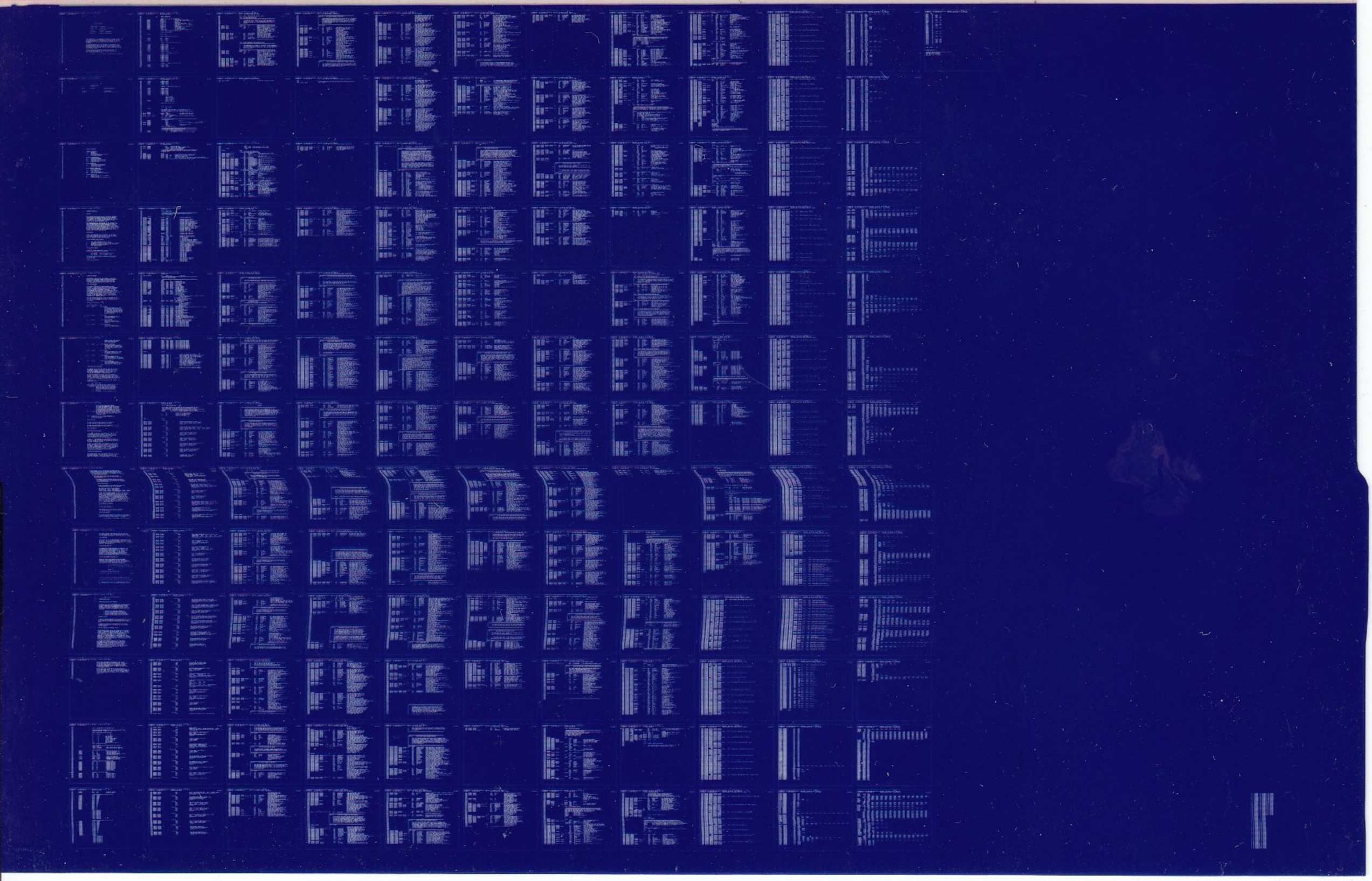
COPYRIGHT © 77-78

FICHE 1 OF 1

AUG 1978

digital

MADE IN USA



```
                        REM     @
```

IDENTIFICATION

PRODUCT CODE	AC-8054B-MC
PRODUCT NAME	CFKTH0 PDP 11/34 MEM MGMT
DATE	JUNE 22 1978
MAINTAINER	DIAGNOSTIC PROGRAMMING
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, 1978 BY DIGITAL EQUIPMENT CORPORATION

```
53
54
55
56
57
58                                        PROGRAM HISTORY
59                                        ---- -------
60
61                                DATE            REVISION            REASON FOR REVISION
62                                - --            -------             --------  ---------
63
64                                31-JAN-77         A                 FIRST RELEASE
65                                28-JUN-78          B                HARDWARE FAULT NOT DETECTED
66
67
```

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
```

```
                                        C       DIGITS TYPED SO FAR TO BE IGNORED
                                                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  S
                        STUCK AT "1" OR IT IS GETTING SET WHEN BIT :4 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

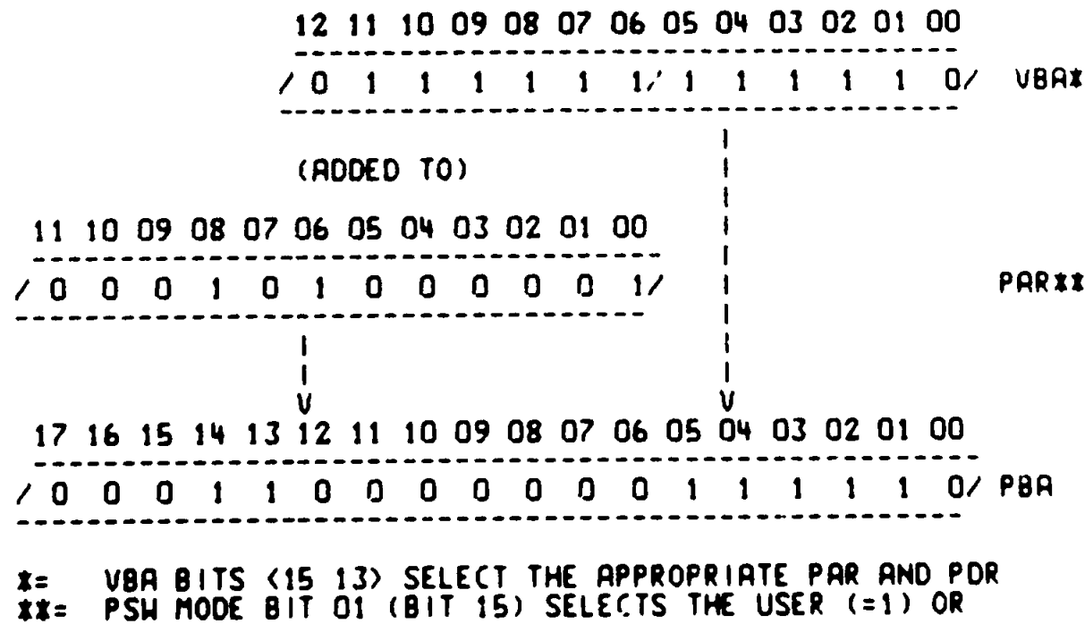
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET
BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS
BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9) T-BIT
TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH
REGISTER (SEE SECTION 2.4)

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
```

SEQ 0011

```
                        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  CFKTH00 PDP 11/34 MEM MGNT DIAG
                               ,*
                               ,*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                               ;*PACKAGE (MAINDEC-11-D20AC-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

172300                          KIPDR0= 172300
172302                          KIPDR1= 172302
172304                          KIPDR2= 172304
```

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

                                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

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

                                         EQUIV  SP,KSP
                                        .EQUIV  SP,USP
                                         EQUIV  BIT4,TBIT
                                         EQUIV  BIT6,WBIT
                        001100          KERSTK= STACK
                        000700          USESTK= STACK-200

                                        ;*ADDITIONAL DEFINITIONS
                                        ;*

                                         SBTTL  TRAP CATCHER

                        000000                  =0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A " +2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
000174  000000         DISPREG: .WORD  0                ;;SOFTWARE DISPLAY REGISTER
000176  000000         SWREG:   .WORD  0                ;;SOFTWARE SWITCH REGISTER
                         SBTTL  STARTING ADDRESS(ES)
000200  000137  020000         JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
                         SBTTL  ACT11 HOOKS

                                        ;;**************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                        000204          $SVPC=                  ;SAVE PC
                        000046           =46
                        034760          $ENDAD                  ;;1)SET LOC 46 TO ADDRESS OF $ENDAD IN  $EOP
                        000052           =52
000052  000000          WORD   0                ;;2)SET LOC 52 TO ZERO
                        000204          .=$SVPC                 ;; RESTORE PC
                         SBTTL  APT PARAMETER BLOCK

                                        ;;**************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;**************************************************************
                        000204          $X=     ;;SAVE CURRENT LOCATION
                        000024          =24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
```

```
000024  000200                  200     ,,FOR APT START UP
        000044                   =44    ,,POINT TO APT INDIRECT ADDRESS PNTR
000044  000204                  $APTHDR ,,POINT TO APT HEADER BLOCK
        000204                   = $X   ,,RESET LOCATION COUNTER
                        ,,**********************************************************
                        ,SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                        ,INTERFACE SPEC

000204                  $APTHD
000204  000000          $HIBTS   WORD   0          ,,TWO HIGH BITS OF 18 BIT MAILBOX ADDR
000206  001226          $MBADR   WORD   $MAIL      ,,ADDRESS OF APT MAILBOX (BITS 0-15)
000210  000010          $TSTM    WORD   10         ,,RUN TIM OF LONGEST TEST
000212  000020          $PASTM   WORD   20         ,,RUN TIME IN SECS  OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000214  000005          $UN TM   WORD   5          ,,ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000216  000052                   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                  $REGAD  WORD    0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000                  $REG0   .WORD   0       ;;CONTAINS (($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
```

```
  810  001224  000012                  $LF       ASCIZ  (12)                ,,LINE FEED
  811                                  ,,*****************************************************************
  812                                   SBTTL  APT MAILBOX-ETABLE
  813
  814                                  ,,*****************************************************************
  815                                   EVEN
  816  001226                          $MAIL                          ,,APT MAILBOX
  817  001226  000000                  $MSGTY    WORD    AMSGTY       ,,MESSAGE TYPE CODE
  818  001230  000000                  $FATAL    WORD    AFATAL       ,,FATAL ERROR NUMBER
  819  001232  000000                  $TESTN    WORD    ATESTN       ,,TEST NUMBER
  820  001234  000000                  $PASS:    WORD    APASS        ,,PASS COUNT
  821  001236  000000                  $DEVCT    WORD    ADEVCT       ,,DEVICE COUNT
  822  001240  000000                  $UNIT.    WORD    AUNIT        ,,I/O UNIT NUMBER
  823  001242  000000                  $MSGAD.   WORD    AMSGAD       ,,MESSAGE ADDRESS
  824  001244  000000                  $MSGLG    WORD    AMSGLG       ,,MESSAGE LENGTH
  825  001246                          $ETABLE                        ,,APT ENVIRONMENT TABLE
  826  001246      000                 $ENV      BYTE    AENV         ,,ENVIRONMENT BYTE
  827  001247      000                 $ENVM:    BYTE    AENVM        ,,ENVIRONMENT MODE BITS
  828  001250  000000                  $SWREG    WORD    ASWREG       ,,APT SWITCH REGISTER
  829  001252  000000                  $USWR:    WORD    AUSWR        ,,USER SWITCHES
  830  001254  000000                  $CPUOP    WORD    ACPUOP       ,,CPU TYPE,OPTIONS
  831                                  ,*                             BITS 15-11=CPU TYPE
  832                                  ,*                                     11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
  833                                  ,*                                     11/70=06,PDQ=07,Q=10
  834                                  ,*                             BIT 10=REAL TIME CLOCK
  835                                  ,*                             BIT  9=FLOATING POINT PROCESSOR
  836                                  ,*                             BIT  8=MEMORY MANAGEMENT
  837  001256      000                 $MAMS1    BYTE    AMAMS1       ,,HIGH ADDRESS,M S  BYTE
  838  001257      000                 $MTYP1    BYTE    AMTYP1       ,,MEM  TYPE,BLK#1
  839                                  ,*                             MEM TYPE BYTE   --  (HIGH BYTE)
  840                                  ,*                                     900 NSEC CORE=001
  841                                  ,*                                     300 NSEC BIPOLAR=002
  842                                  ,*                                     500 NSEC MOS=003
  843  001260  000000                  $MADR1    WORD    AMADR1       ,,HIGH ADDRESS,BLK#1
  844                                  ,*                               MEM LAST ADDR =3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
  845  001262      000                 $MAMS2  . BYTE    AMAMS2       ,,HIGH ADDRESS,M S  BYTE
  846  001263      000                 $MTYP2. . BYTE    AMTYP2       ,,MEM TYPE,BLK#2
  847  001264  000000                  $MADR2  . WORD    AMADR2       ,,MEM LAST ADDRESS,BLK#2
  848  001266      000                 $MAMS3: . BYTE    AMAMS3       ,,HIGH ADDRESS,M S BYTE
  849  001267      000                 $MTYP3: . BYTE    AMTYP3       ,,MEM TYPE,BLK#3
  850  001270  000000                  $MADR3: . WORD    AMADR3       ,,MEM LAST ADDRESS,BLK#3
  851  001272      000                 $MAMS4: . BYTE    AMAMS4       ,;HIGH ADDRESS,M S BYTE
  852  001273      000                 $MTYP4: . BYTE    AMTYP4       ,,MEM TYPE,BLK#4
  853  001274  000000                  $MADR4: . WORD    AMADR4       ,,MEM LAST ADDRESS,BLK#4
  854  001276  000000                  $VECT1: . WORD    AVECT1       ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
  855  001300  000000                  $VECT2. . WORD    AVECT2       ,,INTERRUPT VECTOR#2BUS PRIORITY#2
  856  001302  000000                  $BASE:    WORD    ABASE        ,,BASE ADDRESS OF EQUIPMENT UNDER TEST
  857  001304  000000                  $DEVM:    WORD    ADEVM        ,,DEVICE MAP
  858  001306  000000                  $CDW1.  . WORD    ACDW1        ,,CONTROLLER DESCRIPTION WORD#1
  859  001310  000000                  $CDW2:    WORD    ACDW2        ;,CONTROLLER DESCRIPTION WORD#2
  860  001312  000000                  $DDW0     WORD    ADDW0        ,,DEVICE DESCRIPTOR WORD#0
  861  001314  000000                  $DDW1:  . WORD    ADDW1        ,,DEVICE DESCRIPTOR WORD#1
  862  001316  000000                  $DDW2:    WORD    ADDW2        ,,DEVICE DESCRIPTOR WORD#2
  863  001320  000000                  $DDW3     WORD    ADDW3        ,,DEVICE DESCRIPTOR WORD#3
  864  001322  000000                  $DDW4     WORD    ADDW4        ,,DEVICE DESCRIPTOR WORD#4
  865  001324  000000                  $DDW5     WORD    ADDW5        ,,DEVICE DESCRIPTOR WORD#5
```

```
866  001326  000000          $DDW6   WORD   ADDW6   ,,DEVICE DESCRIPTOR WORD#6
867  001330  000000          $DDW7   WORD   ADDW7   ,,DEVICE DESCRIPTOR WORD#7
868  001332  000000          $DDW8   WORD   ADDW8   ,,DEVICE DESCRIPTOR WORD#8
869  001334  000000          $DDW9   WORD   ADDW9   ,,DEVICE DESCRIPTOR WORD#9
870  001336  000000          $DDW10. WORD   ADDW10  ,,DEVICE DESCRIPTOR WORD#10
871  001340  000000          $DDW11  WORD   ADDW11  ,,DEVICE DESCRIPTOR WORD#11
872  001342  000000          $DDW12  WORD   ADDW12  ,,DEVICE DESCRIPTOR WORD#12
873  001344  000000          $DD  3  WORD   ADDW13  ,,DEVICE DESCRIPTOR WORD#13
874  001346  000000          $DDW14  WORD   ADDW14  ,,DEVICE DESCRIPTOR WORD#14
875  001350  000000          $DDW15  WORD   ADDW15  ,,DEVICE DESCRIPTOR WORD#15
876
877
878  001352                  $ETEND
879
880
881  001352  000000          TESTNO  WORD   0              ,HOLDS TEST NUMBER FOR TYPEOUTS
882  001354  000000          WASR6   WORD   0              ,USED TO STORE THE STACK POINTER AFTER A TRAP
883  001356  000000          TRAPPC  WORD   0              ,USED TO STORE THE PC OF A TRAP OR ABORT
884  001360  000000          TRAPPS: WORD   0              ,USED TO STORE THE PS OF A TRAP OR ABORT
885  001362  000000          CORSR0  WORD   0       ,+USED TO STORE THE CORRECT SR0
886  001364  000000          CORSR2  WORD   0              ,+USED TO STORE THE CORRECT SR2
887  001366  000000          WASSR0. WORD   0              ,USED TO STORE CONTENTS OF SR0
888  001370  000000          WASSR2. WORD   0              ,USED TO STORE CONTENTS OR SR2
889  001372  000000          TBITPS. WORD   0              ,SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
890  001374  000000          ANDADR  WORD   0              ,HOLDS RESULT OF ADDRESSES BEING AND-ED
891  001376  000000          ORADR   WORD   0              ,HOLDS RESULT OF ADDRESSES BEING OR-ED
892  001400  000000          TONUM   WORD   0              ,HOLDS NUMBER OF TIME-OUTS
893  001402  000000          VIRT1   WORD   0              ,HOLDS VIRTUAL ADDRESS TO BE CONVERTED
894  001404  000000          VIRT2   WORD   0              ,   "       "       "
895  001406  000000          PBALO   WORD   0              ,HOLDS BITS <15 00> OF PHYSICAL ADDRESS
896  001410  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 SITEMB.  THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT
                                ;*NOTE1          IF SITEMB 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


001412                          $ERRTB

                                ;*ITEM 1
001412  041616                          EM1             ;UNEXPECTED CPU TRAP TO LOC  004
001414  045620                          DH1             ;OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC
001416  051220                          DT1             ;TRAPPC, TRAPPS, WASR6, TESTNO, $EPPPC, 0
001420  052112                          DF1             ;0,0,0,0,0

                                ;*ITEM 2
001422  041656                          EM2             ;UNEXPECTED MEM  MGMT  TRAP TO LOC  250
001424  045670                          DH2             ;OLD PC  OLD PSW R6 WAS  SR0    SR2    TESTNO  ERRORPC
001426  051234                          DT2             ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
001430  052117                          DF2             ;0,0,0,0,0,0,0

                                ;*ITEM 3
001432  041725                          EM3             ;PRIORITY BITS SET WRONG IN PSW
001434  045760                          DH3             ;WROTE   READ    TESTNO  ERRORPC
001436  051254                          DT3             ;$REG0,$REG1,TESTNO,$ERRPC,0
001440  052126                          DF3             ;0,0,0,0

                                ;*ITEM 4
001442  041764                          EM4             ;MODE BITS SET WRONG IN PSW
001444  045760                          DH3             ;WROTE   READ    TESTNO  ERRORPC
001446  051254                          DT3             ;$REG0,$REG1,TESTNO,$ERRPC,0
001450  052126                          DF3             ;0,0,0,0

                                ;*ITEM 5
001452  042017                          EM5             ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
001454  045760                          DH3             ;WROTE   READ    TESTNO  ERRORPC
001456  051254                          DT3             ;$REG0,$REG1,TESTNO,$ERRPC,0
001460  052126                          DF3             ;0,0,0,0

                                ;*ITEM 6
001462  042072                          EM6             ;KERNEL R6 CHANGED BY WRITING USER R6
001464  045760                          DH3             ;WROTE   READ    TESTNO  ERRORPC
001466  051254                          DT3             ;$REG0,$REG1,TESTNO,$ERRPC,0
001470  052126                          DF3             ;0,0,0,0

                                ;*ITEM 7
001472  042137                          EM7             ;A MEMORY MGMT  REG  TIMED OUT
001474  046020                          DH7             ;ADDRESS TESTNO  ERRORPC
001476  051266                          DT7             ;$REG0,TESTNO,$ERRPC,0
```

```
953  001500  052132                      DF7                     ,0,0,0
954
955                                  ,*ITEM 10
956  001502  042175                      EM10                    ,SUMMARY OF MEM. MGMT  REG  TIMEOUTS
957  001504  046050                      DH10                    ,REGISTER-ADDRS  NUM. OF
958                                                              ,AND-ED  OR-ED   TIMOUTS TESTNO ERRORPC
959  001506  051276                      DT10                    ,ANDADR.ORADR,TONUM,TESTNO,$ERRPC,0
960  001510  052135                      DF10                    ,0,0,1,0,0
961
962                                  ,*ITEM 11
963  001512  042241                      EM11                    ,MEM. MGMT  REG  WOULD NOT CLEAR
964  001514  046150                      DH11                    ,REGISTR READ    READ-(BINARY)
965                                                              ,ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
966  001516  051312                      DT11                    ,$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
967  001520  052142                      DF11                    ,0,0,2,0,0
968
969                                  ,*ITEM 12
970  001522  042301                      EM12                    ,MEM. MGMT  REG  BITS NOT SET CORRECTLY
971  001524  046270                      DH12                    ,REGISTR WROTE   READ    READ
972                                                              ,ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
973  001526  051326                      DT12                    ,$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
974  001530  052147                      DF12                    ,0,0,0,2,0,0
975
976                                  ,*ITEM 13
977  001532  042350                      EM13                    ,SR0 EFFECTED BY WRITE TO PSW
978  001534  046430                      DH13                    ,READ    TESTNO  ERRORPC
979  001536  051344                      DT13                    ,$REG0,TESTNO,$ERRPC,0
980  001540  052155                      DF13                    ,0,0,0
981
982                                  ,*ITEM 14
983  001542  042405                      EM14                    ,SR1 DID NOT READ ALL ZEROS
984  001544  046430                      DH13                    ,READ    TESTNO  ERRORPC
985  001546  051344                      DT13                    ,$REG0,TESTNO,$ERRPC,0
986  001550  052155                      DF13                    ,0,0,0
987
988                                  ,*ITEM 15
989  001552  042440                      EM15                    ,DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
990  001554  046270                      DH12                    ,REGISTER WROTE   READ    READ
991                                                              ,ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO  ERRORPC
992  001556  051326                      DT12                    ,$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
993  001560  052147                      DF12                    ,0,0,0,2,0,0
994
995                                  ,*ITEM 16
996  001562  042514                      EM16                    ,DUAL ADDRESSING BETWEEN PAR-PDR'S
997  001564  046460                      DH16                    ,PAR-PDR PAR-PDR
998                                                              ,CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
999  001566  051354                      DT16                    ,$REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
1000 001570  052160                      DF16                    ,0,0,0,0,0,0
1001
1002                                 ,*ITEM 17
1003 001572  042556                      EM17                    ,PHYS  ADDR  FORMED READ WRONG IN MAINT  MODE
1004 001574  046560                      DH17                    ,PHYSICAL VIRTUAL
1005                                                             ,ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC
1006 001576  051372                      DT17                    ,PBAL0,VIRT1,$REG4,TESTNO,$ERRPC,0
1007 001600  052166                      DF17                    ,3,0,0,0,0
1008
```

```
1009
1010  001602  042626                ,*ITEM 20
                                          EM20          ,PHYS  ADDR  FORMED READ WRONG IN RELOCATE MODE
1011  001604  046650                      DH20          ,PHYSICL PAR 4   PAR 5
1012                                                    ,ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO
1013  001606  051406                      DT20          ,PBALO,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
1014  001610  052173                      DF20          ,3,0,0,0,0,0,0,0
1015
1016                                    ,*ITEM 21
1017  001612  042700                      EM21          ,W-BIT DID NOT GET SET IN PDR
1018  001614  046776                      DH21          ,PDR     VIRTUAL
1019                                                    ,TESTED  ADDRESS TESTNO  ERRORPC
1020  001616  051430                      DT21          ,$REG5,$REG3,TESTNO,$ERRPC,0
1021  001620  052203                      DF21          ,0,0,0,0
1022
1023                                    ,*ITEM 22
1024  001622  042735                      EM22          ,W-BIT SET IN MORE THAN ONE PDR
1025  001624  047056                      DH22          ,PDR IN  PDR     VIRTUAL
1026                                                    ,ERROR   TESTED  ADDRESS TESTNO  ERRORPC
1027  001626  051442                      DT22          ,$REG0,$REG5,$REG3,TESTNO,$ERRPC,0
1028  001630  052207                      DF22          ,0,0,0,0,0
1029
1030                                    ,*ITEM 23
1031  001632  042774                      EM23          ,W-BIT NOT CLEARED BY WRITING TO PDR
1032  001634  047155                      DH23          ,PDR     TESTNO  ERRORPC
1033  001636  051456                      DT23          ,$REG5,TESTNO,$ERRPC,0
1034  001640  052214                      DF23          ,0,0,0
1035
1036                                    ,*ITEM 24
1037  001642  043040                      EM24          ,WRITING SR0 SET W-BIT IN KIPDR7
1038  001644  047205                      DH24          ,PDR WAS EXPECTD TESTNO  ERRORPC
1039  001646  051466                      DT24          ,$REG2,$REG1,TESTNO,$ERRPC,0
1040  001650  052217                      DF24          ,0,0,0,0
1041
1042                                    ,*ITEM 25
1043  001652  043100                      EM25          ,W-BIT GOT SET DURING ODD ADDR  ABORT
1044  001654  047205                      DH24          ,PDR WAS EXPECTD TESTNO  ERRORPC
1045  001656  051466                      DT24          ,$REG2,$REG1,TESTNO,$ERRPC,0
1046  001660  052217                      DF24          ,0,0,0,0
1047
1048                                    ,*ITEM 26
1049  001662  043145                      EM26          ,MEMORY MGMT  ACCESS ABORT DID NOT OCCUR
1050  001664  047245                      DH26          ,PDR 4   PSW     TESTNO  ERRORPC
1051  001666  051500                      DT26          ,$REG2,$TMP0,TESTNO,$ERRPC,0
1052  001670  052217                      DF24          ,0,0,0,0
1053
1054                                    ,*ITEM 27
1055  001672  043215                      EM27          ,ACCESS ERROR DID NOT ABORT INSTRUCTION
1056  001674  047245                      DH26          ,PDR 4   PSW     TESTNO  ERRORPC
1057  001676  051500                      DT26          ,$REG2,$TMP0,TESTNO,$ERRPC,0
1058  001700  052217                      DF24          ,0,0,0,0
1059
1060                                    ,*ITEM 30
1061  001702  043264                      EM30          ,SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
1062  001704  047305                      DH30          ,SR0 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1063  001706  051512                      DT30          ,WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1064  001710  052223                      DF30          ,0,0,0,0,0,0
```

```
1065
1066                                    ,*ITEM 31
1067  001712  043336                            EM31    ,SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR
1068  001714  047365                            DH31    ,SR2 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1069  001716  051530                            DT31    ,WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1070  001720  052223                            DF30    ,0,0,0,0,0,0
1071
1072                                    ,*ITEM 32
1073  001722  043403                            EM32    ,PAGE LGTH  ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1074  001724  047445                            DH32    ,V B A   KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC
1075  001726  051546                            DT32    ,$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1076  001730  052223                            DF30    ,0,0,0,0,0,0
1077
1078                                    ,*ITEM 33
1079  001732  043464                            EM33    ,PAGE LGTH  ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1080  001734  047525                            DH33    ,V B A   KIPDR4  TESTNO  ERRORPC
1081  001736  051564                            DT33    ,$REG0,$REG4,TESTNO,$ERRPC,0
1082  001740  052217                            DF24    ,0,0,0,0
1083
1084                                    ,*ITEM 34
1085  001742  043547                            EM34    ,SR0 DID NOT REPORT PAGE LGTH  ABORT CORRECTLY
1086  001744  047565                            DH34    ,V B A.   KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC
1087  001746  051576                            DT34    ,$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1088  001750  052223                            DF30    ,0,0,0,0,0,0
1089                                    ,*ITEM 35
1090  001752  043336                            EM31    ,SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR
1091  001754  047645                            DH35    ,V B.A.   KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC
1092  001756  051614                            DT35    ,$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1093  001760  052223                            DF30    ,0,0,0,0,0,0
1094
1095                                    ,*ITEM 36
1096  001762  043336                            EM31    ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR
1097  001764  047725                            DH36    ,SR2 WAS EXFECTD TESTNO  ERRORPC
1098  001766  051632                            DT36    ,WASSR2,$REG1,TESTNO,$ERRPC,0
1099  001770  052217                            DF24    ;0,0,0,0
1100
1101                                    ,*ITEM 37
1102  001772  043625                            EM37    ,SR0 OR SR2 CHANGED BY A SECOND ABORT
1103  001774  047765                            DH37    ,FIRST ABORT     SECOND ABORT
1104                                                    ,SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC
1105  001776  051644                            DT37    ,$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1106  002000  052223                            DF30    ,0,0,0,0,0,0
1107
1108                                    ,*ITEM 40
1109  002002  043672                            EM40    ,SR0 OR SR2 WAS NOT "RESET" BY A RESET
1110  002004  050102                            DH40    ,SR0 WAS SR2 WAS TESTNO  ERRORPC
1111  002006  051662                            DT40    ,WASSR0,WASSR2,TESTNO,$ERRPC,0
1112  002010  052217                            DF24    ,0,0,0,0
1113
1114                                    ,*ITEM 41
1115  002012  043741                            EM41    ,SR2 NOT TRACKING CORRECTLY
1116  002014  047725                            DH36    ,SR2 WAS EXPECTD TESTNO  ERROPC
1117  002016  051632                            DT36    ,WASSR2,$REG1,TESTNO,$ERRPC,0
1118  002020  052217                            DF24    ,0,0,0,0
1119
1120                                    ,*ITEM 42
```

```
1121  002022  043774                    EM42            ,DID NOT TRAP THRU KERNEL SPACE
1122  002024  050142                    DH42            ,PSW WAS R6 WAS  TESTNO  ERRORPC
1123  002026  051674                    DT42            ,$REG1,$REG2,TESTNO,$ERRPC,0
1124  002030  052217                    DF24            ,0,0,0,0
1125
1126                                  ,*ITEM 43
1127  002032  044033                    EM43            ,KT ERROR SERVICED ON ODD ADDR  ERROR
1128  002034  047155                    DH23            ,PDR     TESTNO  ERRORPC
1129  002036  051456                    DT23            ,$REG5,TESTNO,$ERRPC,0
1130  002040  052214                    DF23            ,0,0,0
1131
1132                                  ,*ITEM 44
1133  002042  044100                    EM44            ,SR0 OR SR2 CHANGED BY ODD ADDR  ERROR
1134  002044  050202                    DH44            ,EXPECTED        RECEIVED
1135                                                    ;SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC
1136  002046  051706                    DT44            ;$REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1137  002050  052223                    DF30            ,0,0,0,0,0,0
1138
1139                                  ,*ITEM 45
1140  002052  044146                    EM45            ,ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR )
1141  002054  050314                    DH45            ;EXPECTED
1142                                                    ,PSW     PC       SR0     SR2
1143                                                    ;170017  (3$+4)  020147  (3$)
1144                                                    ,RECEIVED
1145                                                    ,PSW     PC       SR0     SR2     TESTNO  ERRORPC
1146  002056  051724                    DT45            ,$REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1147  002060  052223                    DF30            ,0,0,0,0,0,0
1148
1149                                  ,*ITEM 46
1150  002062  044223                    EM46            ,MFPI  INSTRUCTION PUSHED WRONG DATA
1151  002064  050511                    DH46            ,DATA    DATA
1152                                                    ,EXPECTD RECEIVD TESTNO  ERRORPC
1153  002066  051742                    DT46            ,$REG0,$REG1,TESTNO,$ERRPC,0
1154  002070  052231                    DF46            ,0,0,0,0
1155
1156                                  ,*ITEM 47
1157  002072  044266                    EM47            ,MTPI  INSTRUCTION LOADED WRONG DATA
1158  002074  050511                    DH46            ,DATA    DATA
1159                                                    ,EXPECTD RECEIVD TESTNO  ERRORPC
1160  002076  051742                    DT46            ,$REG0,$REG1,TESTNO,$ERRPC,0
1161  002100  052231                    DF46            ,0,0,0,0
1162
1163                                  ,*ITEM 50
1164  002102  044331                    EM50            ,STACK NOT PUSHED BY MFPI-MTPI
1165  002104  050566                    DH50            ,TESTNO  ERRORPC
1166  002106  051754                    DT50            ,TESTNO,$ERRPC,0
1167  002110  052235                    DF50            ,0,0
1168
1169                                  ,*ITEM 51
1170  002112  044367                    EM51            ,KERNEL PAGE ACCESSED INSTEAD OF USER  MFPI-MTPI
1171  002114  050606                    DH51            ,SR0 WAS SR2 WAS TESTNO  ERRORPC
1172  002116  051762                    DT51            ,WASSR0,WASSR2,TESTNO,$ERRPC,0
1173  002120  052237                    DF51            ,0,0,0,0
1174
1175                                  ,*ITEM 52
1176  002122  044445                    EM52            ,WRONG PDR'S REFERENCED WHILE  IN RELOCATE MODE
```

```
1177  002124  050646                DH52                ,PHYSICL PAR 4
1178                                                      ,ADDRESS V.B A   PAR 4   SR0 WAS SR2 WAS PSW    TESTNO
1179  002126  051774                DT52                ,PBALO,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
1180  002130  052243                DF52                ,3,0,0,0,0,0,0,0
1181                              ,*ITEM 53
1182  002132  044523                EM53                ,MFPD INSTRUCTION PUSHED WRONG DATA
1183  002134  050511                DH46                ,DATA    DATA
1184                                                      ,EXPECTD RECEIVD TESTNO  ERRORPC
1185  002136  051742                DT46                ,$REG0,$REG1,TESTNO,$ERRPC,0
1186  002140  052231                DF46                ,0,0,0,0
1187
1188                              ,*ITEM 54
1189  002142  044566                EM54                ,STACK NOT PUSHED BY MFPD-MTPD
1190  002144  050566                DH50                ,TESTNO  ERRORPC
1191  002146  051754                DT50                ,TESTNO,$ERRPC,0
1192  002150  052235                DF50                ,0,0
1193
1194                              ,*ITEM 55
1195  002152  044624                EM55                ,PAR OR PDR WAS CHANGED BY A RESET
1196  002154  046150                DH11                ,REGISTR READ    READ-(BINARY)
1197                                                      ,ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
1198  002156  051312                DT11                ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
1199  002160  052142                DF11                ,0,0,2,0,0
1200
1201                              ,*ITEM 56
1202  002162  044662                EM56                ,ILLEGAL MODE 01 NOT ABORTED
1203  002164  050566                DH50                ;TESTNO  ERRORPC
1204  002166  051754                DT50                ,TESTNO,$ERRPC,0
1205  002170  052235                DF50                ;0,0
1206
1207                              ,*ITEM 57
1208  002172  044716                EM57                ;SR0 DID NOT REPORT ILLEGAL MODE 01 CORRECTLY
1209  002174  050764                DH57                ;SR0 WAS EXPECTD TESTNO  ERRORPC
1210  002176  052016                DT57                ;WASSR0,$REG1,TESTNO,$ERRPC,0
1211  002200  052253                DF57                ;0,0,0,0
1212
1213                              ,*ITEM 60
1214  002202  044773                EM60                ;PSW CHANGED BY AN RTI  IN USER MODE
1215  002204  051024                DH60                ;PSW WAS EXPECTD TESTNO  ERRORPC
1216  002206  052030                DT60                ;$REG1,$REG2,TESTNO,$ERRPC,0
1217  002210  052257                DF60                ;0,0,0,0
1218
1219                              ;*ITEM 61
1220  002212  045036                EM61                ;MAINT MODE (SR0<8>) NOT DISABLED BY A RESET
1221  002214  050566                DH50                ;TESTNO  ERRORPC
1222  002216  051754                DT50                ;TESTNO,$ERRPC,0
1223  002220  052235                DF50                ;0,0
1224
1225                              ,*ITEM 62
1226  002222  045114                EM62                ;DATA INCORRECT AFTER A MAINT  MODE WRITE
1227  002224  045760                DH3                 ;WROTE   READ    TESTNO  ERRORPC
1228  002226  061254                DT3                 ,$REG0,$REG1,TESTNO,$ERRPC,0
1229  002230  052126                DF3                 ,0,0,0,0
1230
1231                              ,*ITEM 63
1232  002232  045165                EM63                ,SOURCE RELOCATED IN MAINT  MODE
```

```
1233  002234  045670                  DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2    TESTNO  ERRORPC
1234  002236  051234                  DT2                     ;TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
1235  002240  052117                  DF2                     ,0,0,0,0,0,0,0
1236
1237                                ,*ITEM 64
1238  002242  043336                  EM31                    ;SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR
1239  002244  047725                  DH36                    ;SR2 WAS EXPECTD TESTNO  ERRORPC
1240  002246  052042                  DT64                    ;WASSR2,$REG4,TESTNO,$ERRPC,0
1241  002250  052217                  DF24                    ,0,0,0,0
1242
1243                                ,*ITEM 65
1244  002252  045225                  EM64                    ,+NON RESIDENT ABORT DID NOT OCCUR
1245  002254  051064                  DH61                    ,+SR0       SR2     TESTNO   ERRORPC
1246  002256  052054                  DT65                    ,+WASSR0,WASSR2,TESTNO,$ERRORPC
1247  002260  052263                  DF61                    ,+0,0,0,0
1248
1249                                ,*ITEM 66
1250  002262  045266                  EM65                    ,+ERROR FLAG FOR KR ABORT DID NOT SET
1     002264  051064                  DH61                    ,+SR0         SR2   TESTNO  ERRORPC
1252  002266  052054                  DT65                    ,+WASSR0,WASSR2,TESTNO,$ERRPC
1253  002270  052263                  DF61                    ,+0,0,0,0
1254
1255
1256                                ;*ITEM67
1257  002272  045351                  EM66                    ,+SR2 DID NOT FREEZE THE VIRTUAL ADRS OF ABORT
1258  002274  051064                  DH61                    ,+SR0       SR2  TESTNO  ERRORPC
1259  002276  052054                  DT65                    ,+WASSR0,WASSR2,TESTNO,$ERRPC
1260  002300  052263                  DF61                    ,+0,0,0,0
1261
1262
1263                                ;*ITEM 70
1264  002302  045441                  EM67                    ,+2ND NON RESIDENT ABORT DID NOT OCCUR
1265  002304  051064                  DH61                    ,+SR0       SR2  TESTNO  ERRORPC
1266  002306  052054                  DT65                    ,+WASSR0,WASSR2,TESTNO,$ERRPC
1267  002310  052263                  DF61                    ,+0,0,0,0
1268
1269                                ,*ITEM 71
1270  002312  045506                  EM70                    ,+2ND NR ABORT CHANGED SR2
1271  002314  051122                  DH62                    ,+SR2 EXPT SR2 RECD TESTNO ERRORPC
1272  002316  052066                  DT66                    ;+CORSR0,WASSR2,TESTNO,$ERRPC
1273  002320  052263                  DF61                    ;+0,0,0,0
1274
1275                                ,*ITEM 72
1276  002322  045562                  EM71                    ,+SR0 WAS NOT CLEARED BY INIT
1277  002324  051161                  DH63                    ,+SR0 EXPT SR0 RECD TESTNO ERRORPC
1278  002326  052100                  DT67                    ,+CORSR0,WASSR0,TESTNO,ERRORPC
1279  002330  052263                  DF61                    ,+0,0,0,0
1280
```

```
1281                                    .SBTTL  ***** TRAP HANDLING ROUTINES *****
1282
1283                                    .SBTTL  CPU TRAP HANDLER ROUTINE
1284                                    ;;*************************************************************************
1285                                    ;*
1286                                    ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
1287                                    ;*      "ERRVEC" (LOC  004)   IF THIS SUBROUTINE IS ENTERED BY A
1288                                    ;*      SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
1289                                    ;*      EXECUTED
1290                                    ;*
1291                                    ;;*************************************************************************
1292  002332  005227                    TIMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
1293  002334  177777                    TIMFLG:  .WORD  -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1294  002336  001401                            BEQ     1$              ;BRANCH IF FIRST TIME IN
1295  002340  000000                            HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
1296                                                                    ;A SECOND TIME BEFORE I FINISHED
1297                                                                    ;REPORTING THE FIRST ERROR   THE
1298                                                                    ;SECOND ENTRY ADDRESS SHOULD BE ON
1299                                                                    ;THE KERNEL STACK
1300  002342  012637  001356            1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
1301  002346  012637  001360                    MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
1302  002352  010637  001354                    MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
1303  002356  104001                            ERROR   1               ;UNEXPECTED TRAP OR ABORT TO LOC  4
1304  002360  012737  177777  002334            MOV     #-1,TIMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1305  002366  013746  001360                    MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
1306  002372  013746  001356                    MOV     TRAPPC,-(KSP)
1307  002376  000006                            RTT                     ;RETURN FROM INTERRUPT OR ABORT
1308
1309
1310                                    .SBTTL  MEMORY MANAGEMENT TRAP HANDLER ROUTINE
1311                                    ;;*************************************************************************
1312                                    ;*
1313                                    ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
1314                                    ;*      TRAPS AND ABORTS THRU "MMVEC" (LOC  250)   IF THIS SUBROUTINE IS
1315                                    ;*      ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
1316                                    ;*      HALT IS EXECUTED.
1317                                    ;*
1318                                    ;;*************************************************************************
1319  002400  005227                    MGMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
1320  002402  177777                    MGMFLG:  .WORD  -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1321  002404  001401                            BEQ     1$              ;BRANCH IF FIRST TIME IN
1322  002406  000000                            HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
1323                                                                    ;A SECOND TIME BEFORE I FINISHED
1324                                                                    ;REPORTING THE FIRST ERROR   THE
1325                                                                    ;SECOND ENTRY ADDRESS SHOULD BE ON
1326                                                                    ;THE KERNEL STACK.
1327  002410  012637  001356            1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
1328  002414  012637  001360                    MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
1329  002420  010637  001354                    MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
1330  002424  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE CONTENTS OF KT STATUS REG  0
1331  002432  013737  177576  001370          MOV     SR2,WASSR2      ;SAVE CONTENTS OF KT STATUS REG  2
1332  002440  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN STATUS REG 0
1333  002446  104002                          ERROR   2               ;UNEXPECTED TRAP OR ABORT TO LOC  250
1334  002450  012737  177777  002402          MOV     #-1,MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1335  002456  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
1336  002462  013746  001356                  MOV     TRAPPC,-(KSP)
```

```
 1337  002466  000006                        RTT                     ,RETURN FROM INTERRUPT OR ABORT
 1338
```

```
                                        SBTTL
                                        SBTTL  *****   STARTING POINT OF TEST  *****
                                        SBTTL  *****   STARTING ADDRESS OF 200 *****
                020000                  =20000

020000                          START
                                 SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
020000  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
020004  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
020006  022706  001140                  CMP     #SWR,R6 ;;DONE?
020012  001374                          BNE      -6             ;;LOOP BACK IF NO
020014  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
020020  012737  035040  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
020026  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
020034  012737  035320  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
020042  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
020050  012737  041302  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
020056  012737  000340  000036          MOV     #340,@#TRAPVEC+2,LEVEL 7
020064  012737  041370  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
020072  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
020100  013737  034606  034600          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
020106  005037  001212                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
020112  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
020116  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
                                ;;INITIALIZE THE "T-BIT" TRAP VECTOR.  THEN LOAD LOCATION "$RTRN", IN
                                ;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT"
020124  012737  035024  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
020132  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
020140  012737  000002  035024          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
020146  012737  020174  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
020154  006046                          CLR     -(SP)           ;;DUMMY PS
020156  012746  020164                  MOV     #64$,-(SP)      ;;AND PC
020162  000006                          RTT                     ;;TRY THE RTT
020164  012737  000006  035024  64$     MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
020172  000402                          BR      66$
020174  062706  000010          65$     ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
020200  012737  000012  000010  66$     MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
020206  005037  035032                  CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
020212  012737  020212  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
020220  012737  020220  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER.  IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER
020226  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
020232  012737  020266  000004          MOV     #67$,@#ERRVEC   ;;SET UP ERROR VECTOR
020240  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
020246  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
020254  022777  177777  160656          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
020262  001012                          BNE     69$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
020264  000403                          BR      68$             ;;BRANCH IF NO TIMEOUT
020266  012716  020274          67$     MOV     #68$,(SP)       ;;SET UP FOR TRAP RETURN
020272  000002                          RTI
020274  012737  000176  001140  68$     MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
020302  012737  000174  001142          MOV     #DISPREG,DISPLAY
```

```
1395  020310  012637  000004          69$     MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1396
1397  020314  005037  001234                  CLR     $PASS           ;;CLEAR PASS COUNT
1398  020320  132737  000200  001247          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
1399  020326  001403                          BEQ     70$             ;;YES, USE NON-APT SWITCH
1400  020330  012737  001250  001140          MOV     #$SWREG,SWR     ;;NO, USE APT SWITCH REGISTER
1401  020336                          70$
1402                                   SBTTL  TYPE PROGRAM NAME
1403                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1404  020336  005227  177777                  INC     #-1             ;;FIRST TIME?
1405  020342  001051                          BNE     71$             ;;BRANCH IF NO
1406  020344  022737  034760  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
1407  020352  001445                          BEQ     71$             ;;BRANCH IF YES
1408  020354  104401  020422                  TYPE    ,72$            ;;TYPE ASCIZ STRING
1409                                   SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1410  020360  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1411  020364  001012                          BNE     73$             ;;BRANCH IF YES
1412  020366  123727  001246  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
1413  020374  001406                          BEQ     73$             ;;BRANCH IF YES
1414  020376  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1415  020404  001005                          BNE     74$             ;;BRANCH IF NO
1416  020406  104407                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1417  020410  000403                          BR      74$
1418  020412  112737  000001  001134  73$     MOVB    #1,$AUTOX       ;;SET AUTO-MODE INDICATOR
1419  020420                          74$
1420  020420  000422                          BR      71$             ;;GET OVER THE ASCIZ
1421                                  ;;72$    ASCIZ   <CRLF>#CFKTHB0  11/34 MEMORY MGMT  DIAG #<CRLF>
1422  020466                          71$
1423
1424  020466                          LOOP
1425  020466  012706  001100                  MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
1426  020472  012737  002332  000004          MOV     #TIMERR,ERRVEC  ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1427  020500  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1428  020506  012737  002400  000250          MOV     #MGMERR,MMVEC   ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1429  020514  012737  000340  000252          MOV     #340,MMVEC+2    ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1430  020522  012700  177777                  MOV     #-1,R0          ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1431  020526  010037  002334                  MOV     R0,TIMFLG       ;INITIALIZE CPU ERROR FLAG
1432  020532  010037  002402                  MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1433  020536  012737  000340  001372          MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1434  020544  005037  177572                  CLR     SR0             ;BE SURE MEM  MGMT IS OFF TO START WITH
1435
```

```
1436
1437
1438                                    ;;*****************************************************************
1439                                    ;*TEST 1        PSW PRIORITY BIT TEST
1440                                    ;*
1441                                    ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD (7 5) "PRIORITY BITS"
1442                                    ;*      TO SEE THAT SOME OF THE BASIC "DATA PATH" LOGIC IS WORKING
1443                                    ;*
1444                                    ;;*****************************************************************
1445  020550  000004                    TST1    SCOPE
1446  020552  012737  020562  001110    1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1447  020560  005000                            CLR     R0              ;INITIALIZE R0 WITH PRIORITY=0 DATA
1448  020562  005001                    2$      CLR     R1              ;PREPARE R1 TO ACCEPT DATA READ
1449  020564  106400                            MTPS    R0              ;WRITE PRIORITY BITS IN THE PSW
1450  020566  106701                            MFPS    R1              ;READ BACK THE LOW BYTE OF PSW
1451  020570  042701  177437                  BIC     #177437,R1      ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
1452  020574  020001                            CMP     R0,R1           ;WAS CORRECT PRIORITY SET IN THE PSW?
1453  020576  001401                            BEQ     3$              ;BRANCH IF YES
1454  020600  104003                            ERROR   3               ;PRIORITY BITS SET WRONG IN PSW
1455                                                                    ;FOR TIGHTER SCOPE LOOP
1456                                                                    ;REPLACE ERROR CALL WITH
1457                                                                    ;"BR 2$" = 000770
1458  020602  062700  000040            3$      ADD     #40,R0          ;CHANGE DATA TO NEXT PRIORITY
1459  020606  022700  000400                    CMP     #400,R0         ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
1460  020612  001363                            BNE     2$              ;BRANCH IF NO
1461  020614  012737  020552  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1462
1463                                    ;;*****************************************************************
1464                                    ;*TEST 2        PSW MODE BIT TEST
1465                                    ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD (15 12) 'MODE BITS"
1466                                    ;*      TO FURTHER CHECK THE BASIC CPU DATA PATHS
1467                                    ;*
1468                                    ;;*****************************************************************
1469  020622  000004                    TST2    SCOPE
1470  020624  012737  020634  001110    1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1471  020632  005000                            CLR     R0              ;INITIALIZE R0 WITH MODE BITS = 0000
1472  020634  005037  177776            2$      CLR     PSW             ;INITIALIZE PSW
1473  020640  050037  177776                    BIS     R0,PSW          ;BIT SET THE PSW MODE BITS WITH R0
1474  020644  013701  177776                    MOV     PSW,R1          ;READ BACK THE CONTENTS OF THE PSW
1475  020650  042701  007777                    BIC     #007777,R1      ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
1476  020654  020001                            CMP     R0,R1           ;WERE THE MODE BITS SET CORRECTLY?
1477  020656  001403                            BEQ     3$              ;BRANCH IF YES
1478  020660  005037  177776                    CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
1479  020664  104004                            ERROR   4               ;MODE BITS SET WRONG IN PSW
1480                                                                    ;FOR TIGHTER SCOPE LOOP
1481                                                                    ;REPLACE ERROR CALL WITH
1482                                                                    ;"BR 2$" = 000763
1483  020666  062700  010000            3$      ADD     #10000,R0       ;CHANGE MODE BIT DATA
1484  020672  001360                            BNE     2$              ;BRANCH IF STILL MORE COMBINATIONS
1485  020674  012737  020624  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1486  020702  005037  177776                    CLR     PSW             ;RESET PSW BEFORE LEAVING
1487
1488                                    ;;*****************************************************************
1489                                    ;*TEST 3        BYTE ADDRESSING TEST FOR PSW
1490                                    ;*
1491                                    ;*      THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
```

```
1492                                    ;*      AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY
1493                                    ;*      THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC
1494                                    ;*
1495                                    ;;**********************************************************************
1496  020706  000004                    TST3    SCOPE
1497  020710  012737  020716  001110    1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1498  020716  005037  177776            2$      CLR     PSW             ;CLEAR THE PSW
1499  020722  012700  000360                    MOV     #360,R0         ;PUT THE HIGH BYTE DATA INTO R0
1500  020726  110037  177777                    MOVB    R0,PSW+1        ;WRITE THE HIGH BYTE OF THE PSW
1501  020732  013701  177776                    MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
1502  020736  042701  007437                    BIC     #007437,R1      ;MASK OFF THE T & CC BITS
1503  020742  000300                            SWAB    R0              ;GET DATA WRITTEN IN HIGH BYTE OF R0
1504  020744  020001                            CMP     R0,R1           ;WAS THE PSW WRITTEN TO CORRECTLY
1505  020746  001403                            BEQ     3$              ;BRANCH IF YES
1506  020750  005037  177776                    CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
1507  020754  104005                            ERROR   5               ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
1508                                                                    ;FOR TIGHTER SCOPE LOOP
1509                                                                    ;REPLACE ERROR CALL WITH
1510                                                                    ;"BR 2$" = 000760
1511  020756  012737  020764  001110    3$      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
1512  020764  005037  177776            4$      CLR     PSW             ;CLEAR THE PSW
1513  020770  012700  000340                    MOV     #340,R0         ;PUT THE LOW BYTE DATA INTO R0
1514  020774  110037  177776                    MOVB    R0,PSW          ;WRITE THE LOW BYTE OF THE PSW
1515  021000  013701  177776                    MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
1516  021004  042701  007437                    BIC     #007437,R1      ;MASK OFF THE T&CC BITS
1517  021010  020001                            CMP     R0,R1           ;WAS PSW WRITTEN TO CORRECTLY
1518  021012  001403                            BEQ     5$              ;BRANCH IF YES
1519  021014  005037  177776                    CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
1520  021020  104005                            ERROR   5               ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
1521                                                                    ;FOR TIGHTER SCOPE LOOP
1522                                                                    ;REPLACE ERROR CALL WITH
1523                                                                    ;"BR 2$" = 000736
1524  021022  012737  020710  001110    5$      MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1525
1526                                    ;;**********************************************************************
1527                                    ;*TEST 4         TEST AND SETUP OF STACK POINTERS
1528                                    ;*
1529                                    ;*      THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
1530                                    ;*      REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
1531                                    ;*      EACH OTHER   KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
1532                                    ;*      KERNEL R6 IS READ TO BE SURE ITS STILL 1100
1533                                    ;*
1534                                    ;;**********************************************************************
1535  021030  000004                    TST4    SCOPE
1536  021032  005037  177776                    CLR     PSW             ;GO TO KERNEL MODE
1537  021036  012706  001100                    MOV     #KERSTK,KSP     ;SET KERNEL STACK POINTER TO 1100
1538  021042  012737  140000  177776            MOV     #140000,PSW     ;GO TO USER MODE
1539  021050  012706  000700                    MOV     #USESTK,USP     ;SET USER STACK POINTER TO 700
1540  021054  005037  177776                    CLR     PSW             ;BACK TO KERNEL MODE
1541  021060  022706  001100                    CMP     #KERSTK,KSP     ;IS KERNEL R6 STILL 1100?
1542  021064  001404                            BEQ     TST5            ;;BRANCH IF KERNEL R6 IS OKAY
1543  021066  012700  001100                    MOV     #KERSTK,R0      ;SAVE DATA WRITTEN FOR ERROR REPORT
1544  021072  010601                            MOV     KSP,R1          ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
1545  021074  104006                            ERROR   6               ;KERNEL R6 CHANGED BY WRITING USER R6
1546                                                                    ;FOR TIGHTER SCOPE LOOP
1547                                                                    ;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.
                                ;*
                                ;;*****************************************************************
021076  000004                TST5    SCOPE

021100  012737  021142  001110  1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
021106  012737  021200  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
021114  012700  177572                  MOV     #SR0,R0         ;LOAD R0 WITH ADDRESS OF FIRST REG
021120  012701  000003                  MOV     #3,R1           ;LOAD R1 WITH THE LOOP COUNT
021124  012737  177777  001374          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDRS. LOC
021132  005037  001376                  CLR     ORADR           ;INITIALIZE "OR" OF ADDRS. LOC
021136  005037  001400                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
021142  005710                  2$      TST     (R0)            ;TRY ADDRESSING A STATUS REGISTER
                                                                ;IF IT TIMES OUT GO TO 5$
021144  062700  000002          3$      ADD     #2,R0           ;PUT NEXT ADDRESS IN R0
021150  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
021152  012737  021100  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
021160  005737  001400                  TST     TONUM           ;DID ANY OF THE STATUS REG.S TIMEOUT?
021164  001401                          BEQ     4$              ;BRANCH IF NO
021166  104010                          ERROR   10              ;SUMMARY OF STATUS REG. TIMEOUTS
021170  012737  002332  000004  4$      MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021176  000414                          BR      TST6            ;;GO TO NEXT TEST

021200  062706  000004          5$      ADD     #4,KSP          ;CLEAN UP THE STACK
021204  104007                          ERROR   7               ;ONE OF THE STATUS REGS  TIMED OUT
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000756
021206  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021210  050237  001376                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS  THAT TIMED OUT
021214  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS  THAT TIMED OUT
021216  040237  001374                  BIC     R2,ANDADR
021222  005237  001400                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
```

```
021226 000746                               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.
                                    ,*
                                    ,,************************************************************
021230 000004                       TST6    SCOPE

021232 012737 021274 001110         1$      MOV     #2$,$LPERR      ,SET LOOP ON ERROR POINTER TO 2$
021240 012737 021332 000004                 MOV     #5$,@#4         ,SET TIMEOUT VECTOR TO 5$
021246 012700 172340                        MOV     #KIPAR0,R0      ,LOAD R0 WITH ADDRESS OF FIRST REG
021252 012701 000010                        MOV     #10,R1          ,LOAD R1 WITH LOOP COUNT (8)
021256 012737 177777 001374                 MOV     #-1,ANDADR      ,INITIALIZE "AND" OF ADDR. LOC
021264 005037 001376                        CLR     ORADR           ,INITIALIZE "OR" OF ADDR. LOC
021270 005037 001400                        CLR     TONUM           ,INITIALIZE "TIMEOUTS" COUNTER
021274 005710                       2$      TST     (R0)            ,TRY ADDRESSING A KIPAR
                                                                    ,IF IT TIMES OUT, WILL GO TO 5$
021276 062700 000002                3$      ADD     #2,R0           ,PUT NEXT KIPAR ADDRESS IN R0
021302 077104                               SOB     R1,2$           ,LOOP BACK TO 2$ UNTIL ALL TESTED
021304 012737 021232 001110                 MOV     #1$,$LPERR      ,RESET LOOP ON ERROR POINTER TO 1$
021312 005737 001400                        TST     TONUM           ,DID ANY OF THE KIPARS TIME OUT?
021316 001401                               BEQ     4$              ,BRANCH IF NO
021320 104010                               ERROR   10              ,SUMMARY OF KIPAR TIMEOUTS
021322 012737 002332 000004         4$      MOV     #TIMERR,@#4     ,RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021330 000414                               BR      TST7            ,,GO TO NEXT TEST

021332 062706 000004                5$      ADD     #4,KSP          ,CLEAN UP THE STACK
021336 104007                               ERROR   7               ,ONE OF THE KIPARS TIMED OUT
                                                                    ,FOR TIGHTER SCOPE LOOP
                                                                    ,REPLACE ERROR CALL WITH
                                                                    ,"BR 2$" = 000756
021340 010002                               MOV     R0,R2           ,LOAD THE ADDRESS THAT TIMED OUT INTO R2
021342 050237 001376                        BIS     R2,ORADR        ,"OR" IT WITH OTHER ADDRS  THAT TIMED OUT
021346 005102                               COM     R2              ,"AND" IT WITH OTHER ADDRS  THAT TIMED OUT
021350 040237 001374                        BIC     R2,ANDADR
021354 005237 001400                        INC     TONUM           ,INCREMENT THE TIMEOUT COUNTER
021360 000746                               BR      3$              ,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
                                    ,*
                                    ,,************************************************************
021362 000004                       TST7:   SCOPE

021364 012737 021426 001110         1$      MOV     #2$,$LPERR      ,SET LOOP ON ERROR POINTER TO 2$
021372 012737 021464 000004                 MOV     #5$,@#4         ,SET TIMEOUT VECTOR TO 5$
021400 012700 172300                        MOV     #KIPDR0,R0      ,LOAD R0 WITH ADDRESS OF FIRST REG
021404 012701 000010                        MOV     #10,R1          ,LOAD R1 WITH LOOP COUNT (8)
```

```
1660  021410  012737  177777  001374          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR  LOC
1661  021416  005037  001376                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR.  LOC
1662  021422  005037  001400                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1663  021426  005710                  2$      TST     (R0)            ;TRY ADDRESSING A KIPDR
1664                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1665  021430  062700  000002          3$      ADD     #2,R0           ;PUT NEXT KIPDR ADDRESS IN R0
1666  021434  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1667  021436  012737  021364  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1668  021444  005737  001400                  TST     TONUM           ;DID ANY OF THE KIPDRS TIME OUT?
1669  021450  001401                          BEQ     4$              ;BRANCH IF NO
1670  021452  104010                          ERROR   10              ;SUMMARY OF KIPDR TIMEOUTS
1671  021454  012737  002332  000004  4$      MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1672  021462  000414                          BR      TST10           ;;GO TO NEXT TEST
1673
1674  021464  062706  000004          5$      ADD     #4,KSP          ;CLEAN UP THE STACK
1675  021470  104007                          ERROR   7               ;ONE OF THE KIPDRS TIMED OUT
1676                                                                  ;FOR TIGHTER SCOPE LOOP
1677                                                                  ;REPLACE ERROR CALL WITH
1678                                                                  ;"BR 2$" = 000756
1679  021472  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1680  021474  050237  001376                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS  THAT TIMED OUT
1681  021500  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS  THAT TIMED OUT
1682  021502  040237  001374                  BIC     R2,ANDADR
1683  021506  005237  001400                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
1684  021512  000746                          BR      3$              ;BRANCH BACK TO TEST THE NEXT KIPDR
1685
1686                                  ;;*****************************************************************
1687                                  ;*TEST 10       USER PAR'S TIMEOUT TEST
1688                                  ;*
1689                                  ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1690                                  ;*      REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1691                                  ;*      RESPONDS TO THEIR ADDRESSES
1692                                  ;*
1693                                  ;;*****************************************************************
1694  021514  000004          TST10   SCOPE
1695
1696  021516  012737  021560  001110  1$.     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1697  021524  012737  021616  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
1698  021532  012700  177640                  MOV     #UIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG
1699  021536  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
1700  021542  012737  177777  001374          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR.  LOC
1701  021550  005037  001376                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR  LOC
1702  021554  005037  001400                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1703  021560  005710                  2$      TST     (R0)            ;TRY ADDRESSING A UIPAR
1704                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1705  021562  062700  000002          3$      ADD     #2,R0           ;PUT NEXT UIPAR ADDRESS IN R0
1706  021566  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1707  021570  012737  021516  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1708  021576  005737  001400                  TST     TONUM           ;DID ANY OF THE UIPARS TIME OUT?
1709  021602  001401                          BEQ     4$              ;BRANCH IF NO
1710  021604  104010                          ERROR   10              ;SUMMARY OF UIPAR TIMEOUTS
1711  021606  012737  002332  000004  4$      MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1712  021614  000414                          BR      TST11           ;;GO TO NEXT TEST
1713
1714  021616  062706  000004          5$      ADD     #4,KSP          ;CLEAN UP THE STACK
1715  021622  104007                          ERROR   7               ;ONE OF THE UIPARS TIMED OUT
```

```
1716                                                    ,FOR TIGHTER SCOPE LOOP
1717                                                    ,REPLACE ERROR CALL WITH
1718                                                    ,"BR 2$" = 000756
1719  021624  010002                    MOV     R0,R2   ,LOAD THE ADDRESS THAT TIMED OUT INTO R2
1720  021626  050237  001376            BIS     R2,ORADR        ,"OR" IT WITH OTHER ADDRS  THAT TIMED OUT
1721  021632  005102                    COM     R2              ,"AND" IT WITH OTHER ADDRS  THAT TIMED OUT
1722  021634  040237  001374            BIC     R2,ANDADR
1723  021640  005237  001400            INC     TONUM           ,INCREMENT THE TIMEOUT COUNTER
1724  021644  000746                    BR      3$              ,BRANCH BACK TO TEST THE NEXT UIPAR
1725
1726                                ;,**********************************************************************
1727                                ;*TEST 11        USER PDR'S TIMEOUT TEST
1728                                ,*
1729                                ,*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
1730                                ,*      REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
1731                                ;*      RESPONDS TO THEIR ADDRESSES
1732                                ,*
1733                                ,,**********************************************************************
1734  021646  000004                TST11.  SCOPE
1735
1736  021650  012737  021712  001110  1$.   MOV     #2$,$LPERR      ,SET LOOP ON ERROR POINTER TO 2$
1737  021656  012737  021750  000004        MOV     #5$,@#4         ,SET TIMEOUT VECTOR TO 5$
1738  021664  012700  177600                MOV     #UIPDR0,R0      ,LOAD R0 WITH ADDRESS OF FIRST REG
1739  021670  012701  000010                MOV     #10,R1          ,LOAD R1 WITH LOOP COUNT (8)
1740  021674  012737  177777  001374        MOV     #-1,ANDADR      ,INITIALIZE "AND" OF ADDR  LOC
1741  021702  005037  001376                CLR     ORADR           ,INITIALIZE "OR" OF ADDR. LOC
1742  021706  005037  001400                CLR     TONUM           ,INITIALIZE "TIMEOUTS" COUNTER
1743  021712  005710                2$      TST     (R0)            ,TRY ADDRESSING A UIPDR
1744                                                                ,IF IT TIMES OUT, WILL GO TO 5$
1745  021714  062700  000002        3$      ADD     #2,R0           ,PUT NEXT UIPDR ADDRESS IN R0
1746  021720  077104                        SOB     R1,2$           ,LOOP BACK TO 2$ UNTIL ALL TESTED
1747  021722  012737  021650  001110        MOV     #1$,$LPERR      ,RESET LOOP ON ERROR POINTER TO 1$
1748  021730  005737  001400                TST     TONUM           ,DID ANY OF THE UIPDRS TIME OUT?
1749  021734  001401                        BEQ     4$              ,BRANCH IF NO
1750  021736  104010                        ERROR   10              ,SUMMARY OF UIPDR TIMEOUTS
1751  021740  012737  002332  000004  4$    MOV     #TIMERR,@#4     ,RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1752  021746  000414                        BR      TST12           ,,GO TO NEXT TEST
1753
1754  021750  062706  000004        5$      ADD     #4,KSP          ,CLEAN UP THE STACK
1755  021754  104007                        ERROR   7               ,ONE OF THE UIPDRS TIMED OUT
1756                                                                ,FOR TIGHTER SCOPE LOOP
1757                                                                ,REPLACE ERROR CALL WITH
1758                                                                ,"BR 2$" = 000756
1759  021756  010002                        MOV     R0,R2           ,LOAD THE ADDRESS THAT TIMED OUT INTO R2
1760  021760  050237  001376                BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS  THAT TIMED OUT
1761  021764  005102                        COM     R2              ,"AND" IT WITH OTHER ADDRS  THAT TIMED OUT
1762  021766  040237  001374                BIC     R2,ANDADR
1763  021772  005237  001400                INC     TONUM           ,INCREMENT THE TIMEOUT COUNTER
1764  021776  000746                        BR      3$              ;BRANCH BACK TO TEST THE NEXT UIPDR
1765
1766                                ,,**********************************************************************
1767                                ,*TEST 12        SR0(15:13) BIT TEST & SR2 TEST
1768                                ,*
1769                                ,*      THIS TEST CHECKS BITS (15:13) OF STATUS REGISTER 0 TO SEE
1770                                ,*      THAT EACH CAN BE SET AND CLEARED AND THAT A "RESET" WILL
1771                                ,*      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
                                        ;
                                        ;;*****************************************************************
022000  000004                        TST12   SCOPE

022002  012700  177572                1$      MOV     #SR0,R0         ;LOAD ADDRESS OF SR0 INTO R0
022006  012710  160000                        MOV     #160000,(R0)    ;SET BITS <15 13> IN SR0 (ERROR BITS)
022012  000005                                RESET                   ;ISSUE AND "INIT" SIGNAL
022014  011001                                MOV     (R0),R1         ;READ SR0 INTO R1 TO SEE IF CLEAR
022016  001404                                BEQ     2$              ;BRANCH IF SR0<15:13> CLEARED BY "INIT"
022020  104011                                ERROR   11              ;SR0<15:13> NOT CLEARED BY A "RESET"
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 1$" = 000770
022022  012737  022030  001110                MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
022030  013737  177576  001370        2$      MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
022036  012701  022030                        MOV     #2$,R1          ;LOAD EXPECTED CONTENTS INTO R1
022042  020137  001370                        CMP     R1,WASSR2       ;IS SR2 TRACKING?
022046  001401                                BEQ     3$              ;BRANCH IF YES
022050  104041                                ERROR   41              ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 2$" = 000767
022052  012737  022070  001110        3$      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
022060  012701  100000                        MOV     #BIT15,R1       ;PUT DATA TO BE WRITTEN IN R1
022064  012703  000003                        MOV     #3,R3           ;SETUP R3 AS A LOOP COUNTER
022070  005010                        4$      CLR     (R0)            ;CLEAR SR0
022072  050110                        5$      BIS     R1,(R0)         ;SET ONE OF THE ERROR BITS IN SR0
022074  011002                                MOV     (R0),R2         ;READ SR0 INTO R2
022076  020102                                CMP     R1,R2           ;DID RIGHT ERROR BIT GET SET?
022100  001401                                BEQ     6$              ;BRANCH IF YES
022102  104012                                ERROR   12              ;BITS WERE SET WRONG IN SR0
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 4$" = 000772
022104  012704  022072                6$      MOV     #5$,R4          ;LOAD EXPECTED CONTENTS OF SR2 IN R4
022110  013737  177576  001370                MOV     SR2,WASSR2      ;READ SR2
022116  020437  001370                        CMP     R4,WASSR2       ;DID SR2 LOCK UP WHEN ERROR
                                                                        ;BIT SET IN SR1?
022122  001401                                BEQ     7$              ;BRANCH IF YES
022124  104064                                ERROR   64              ;SR2 DID NOT LOCK UP
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 4$" = 000761
022126  006001                        7$      ROR     R1              ;CHANGE DATA TO CHECK NEXT ERROR BIT
022130  077321                                SOB     R3,4$           ;LOOP BACK UNTIL <15:13> ALL TESTED
022132  005010                                CLR     (R0)            ;CLEAR SR0 BEFORE LEAVING
022134  012737  022002  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
                                ;*
                                ;;*****************************************************************
022142  000004                TST13   SCOPE
                                
022144  005037  177776        1$      CLR     PSW             ;CLEAR THE PSW
022150  005037  177572                CLR     SR0             ;CLEAR STATUS REGISTER 0
022154  106427  000340                MTPS    #340            ;SET PRIORITY 7 IN LOW BYTE OF PSW
022160  013700  177572                MOV     SR0,R0          ;READ STATUS REGISTER 0
022164  001401                        BEQ     2$              ;BRANCH IF IT WAS STILL 0
022166  104013                        ERROR   13              ;SR0 EFFECTED BY A WRITE TO THE PSW
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 1$" = 000767
022170  005037  177572        2$      CLR     SR0             ;BE SURE SR0 IS 0 BEFORE LEAVING
022174  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
                                ;*
                                ;;*****************************************************************
022200  000004                TST14:  SCOPE
022202  012700  177777                MOV     #-1,R0          ;FILL R0 WITH ALL ONES
022206  013700  177574                MOV     SR1,R0          ;READ SR1 INTO R0
022212  001401                        BEQ     TST15           ;;BRANCH IF SR1 READS ALL ZEROS
022214  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.
                                ;*
                                ;;*****************************************************************
022216  000004                TST15:  SCOPE
                                
022220  012700  172340        1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR IN R0
022224  012703  000010        2$      MOV     #10,R3          ;SETUP R3 TO COUNT 8 PAR'S
022230  012737  022236  001110        MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022236  005010                3$      CLR     (R0)            ;CLEAR THE PAR
022240  011001                        MOV     (R0),R1         ;READ THE PAR INTO R1
022242  001401                        BEQ     4$              ;BRANCH IF PAR CLEARED OK
022244  104011                        ERROR   11              ;PAR WOULD NOT CLEAR
```

```
1884                                                            ;FOR TIGHTER SCOPE LOOP
1885                                                            ;REPLACE ERROR CALL WITH
1886                                                            ;"BR 3$" = 000774
1887  022246  012704  167777          4$      MOV     #167777,R4      ;LOAD "WALKING 0" TEST PATTERN IN R4
1888  022252  012737  022260  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1889  022260  005010                  5$      CLR     (R0)            ;CLEAR THE PAR BEFORE LOADING DATA
1890  022262  010401                          MOV     R4,R1           ;LOAD DATA INTO R1
1891  022264  042701  170000                  BIC     #170000,R1      ;MASK UNUSED BITS OUT OF THE DATA
1892  022270  050110                          BIS     R1,(R0)         ;BIT SET THE TEST PATTERN INTO THE PAR
1893  022272  011002                          MOV     (R0),R2         ;READ THE PAR INTO R2
1894  022274  020102                          CMP     R1,R2           ;DOES DATA WRITTEN=DATA READ?
1895  022276  001401                          BEQ     6$              ;BRANCH IF YES
1896  022300  104012                          ERROR   12              ;PAR BITS DID NOT SET CORRECTLY
1897                                                            ;FOR TIGHTER SCOPE LOOP
1898                                                            ;REPLACE ERROR CALL WITH
1899                                                            ;"BR 5$" = 000767
1900  022302  000261                  6$      SEC                     ;SET THE C-BIT FOR THE ROTATE INST
1901  022304  006004                          ROR     R4              ;ROTATE THE TEST PATTERN IN R4
1902  022306  103764                          BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
1903  022310  062700  000002                  ADD     #2,R0           ;GET NEXT PAR ADDRESS IN R0
1904  022314  077330                          SOB     R3,3$           ;BRANCH BACK UNTIL ALL PAR'S TESTED
1905  022316  022700  177660                  CMP     #UIPAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
1906  022322  103003                          BHIS    7$              ;BRANCH IF YES
1907  022324  012700  177640                  MOV     #UIPAR0,R0      ;LOAD FIRST USER PAR ADDR. IN R0
1908  022330  000735                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
1909  022332  012737  022220  001110  7$      MOV     #1$,$LPERR      ;RESET LOOP OR ERROR POINTER TO 1$
1910                                                            ;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.
                                        ;*
                                        ;*****************************************************************
022340  000004                  TST16   SCOPE
022342  012700  172300          1$      MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST PDR IN R0
022346  012703  000010          2$      MOV     #10,R3          ;SETUP R3 TO COUNT 8 PDR'S
022352  012737  022360  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022360  005010                  3$      CLR     (R0)            ;CLEAR THE PDR
022362  011001                          MOV     (R0),R1         ;READ THE PDR INTO R1
022364  001401                          BEQ     4$              ;BRANCH IF PDR CLEARED OK
022366  104011                          ERROR   11              ;PDR WOULD NOT CLEAR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000774
022370  012704  077777          4$      MOV     #077777,R4      ;LOAD "WALKING "0" TEST PATTERN IN R4
022374  012737  022402  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
022402  005010                  5$      CLR     (R0)            ;CLEAR THE PDR BEFORE LOADING DATA
022404  010401                          MOV     R4,R1           ;LOAD DATA INTO R1
022406  042701  100361                  BIC     #100361,R1      ;MASK UNUSED BITS OUT OF THE DATA
022412  050110                          BIS     R1,(R0)         ;BIT SET THE TEST PATTERN INTO THE PDR
022414  011002                          MOV     (R0),R2         ;READ THE PDR :NTO R2
022416  020102                          CMP     R1,R2           ;DOES DATA WRITTEN=DATA READ?
022420  001401                          BEQ     6$              ;BRANCH IF YES
022422  104012                          ERROR   12              ;PDR BITS DID NOT SET CORRECTLY
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000767
022424  000261                  6$      SEC                     ;SET THE C-BIT FOR THE ROTATE INST
022426  006004                          ROR     R4              ;ROTATE THE TEST PATTERN IN R4
022430  103764                          BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
022432  062700  000002                  ADD     #2,R0           ;GET NEXT PDR ADDRESS IN R0
022436  077330                          SOB     R3,3$           ;BRANCH BACK UNTIL ALL PDR'S TESTED
022440  022700  177620                  CMP     #UIPDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
022444  103003                          BHIS    7$              ;BRANCH IF YES
022446  012700  177600                  MOV     #UIPDR0,R0      ;LOAD FIRST USER PDR ADDR  IN R0
022452  000735                          BR      2$              ;BRANCH BACK TO TEST USER PDR'S
022454  012737  022342  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
```

```
                                    ;*
                                    ;;**********************************************************************
022462  000004                      TST17   SCOPE
```

```
1970
1971  022464  012700  172340          1$      MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR INTO R0
1972  022470  012737  022502  001110  2$      MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1973  022476  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER TO DO 8 PAR'S
1974  022502  012701  177777          3$      MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
1975  022506  005010                          CLR     (R0)            ;CLEAR THE PAR
```

```
1976  022510  110110                          MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1977  022512  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1978  022514  042701  177400                  BIC     #177400,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1979  022520  020102                          CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO
1980  022522  001401                          BEQ     4$              ;BRANCH IF YES
1981  022524  104015                          ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1982                                                                  ;FOR TIGHTER SCOPE LOOP
1983                                                                  ;REPLACE ERROR CALL WITH
1984                                                                  ;"BR 3$" = 000766
1985  022526  012737  022534  001110  4$      MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1986  022534  005010                  5$      CLR     (R0)            ;CLEAR THE PAR
1987  022536  012701  177777                  MOV     #-1,R1          ;LOAD TEST, PATTERN INTO R1
1988  022542  110160  000001                  MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1989  022546  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1990  022550  042701  170377                  BIC     #170377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
1991  022554  020102                          CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1992  022556  001401                          BEQ     6$              ;BRANCH IF YES
1993  022560  104015                          ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1994                                                                  ;FOR TIGHTER SCOPE LOOP
1995                                                                  ;REPLACE ERROR CALL WITH
1996                                                                  ;"BR 5" = 000765
1997  022562  062700  000002          6$      ADD     #2,R0           ;PUT ADDRESS OF NEXT PAR IN R0
1998  022566  077333                          SOB     R3,3$           ;BRANCH BACK UNTIL 8 PAR'S TESTED
1999  022570  022700  177660                  CMP     #UIPAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
2000  022574  103003                          BHIS    7$              ;BRANCH IF YES
2001  022576  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2002  022602  000732                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
2003  022604  012737  022464  001110  7$      MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2004
```

```
                                        ;;*************************************
                                        ;*TEST 20       TEST FOR DUAL BYTE AD  ESSING 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
                                        ;*
                                        ;;*************************************************************************
022612  000004                  TST20   SCOPE

022614  012700  172300          1$      MOV     #KIPD%0,R0      ;LOAD ADDRESS OF FIRST PDR INTO R0
022620  012737  022632  001110  2$      MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022626  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER TO DO 8 PDR'S
022632  012701  177777          3$      MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022636  005010                          CLR     (R0)            ;CLEAR THE PDR
022640  110110                          MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PDR
022642  011002                          MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022644  042701  177761                  BIC     #177761,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
022650  020102                          CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO?
022652  001401                          BEQ     4$              ;BRANCH IF YES
022654  104015                          ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000766
022656  012737  022664  001110  4$      MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
022664  005010                  5$      CLR     (R0)            ;CLEAR THE PDR
022666  012701  177777                  MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022672  110160  000001                  MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
022676  011002                          MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022700  042701  100377                  BIC     #100377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
022704  020102                          CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
022706  001401                          BEQ     6$              ;BRANCH IF YES
022710  104015                          ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000765
022712  062700  000002          6$      ADD     #2,R0           ;PUT ADDRESS OF NEXT PDR IN R0
022716  077333                          SOB     R3,3$           ;BRANCH BACK UNTIL 8 PDR'S TESTED
022720  022700  177620                  CMP     #UIPDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
022724  103003                          BHIS    7$              ;BRANCH IF YES
022726  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
022732  000732                          BR      2$              ;BRANCH BACK TO TEST USER PDR'S
022734  012737  022614  001110  7$      MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

```

```
2053                                    ;;*****************************************************************
2054                                    ;*TEST 21        PAR-PDR DUAL ADDRESSING TEST
2055                                    ;*
2056                                    ;*      THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
2057                                    ;*      IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
2058                                    ;*      SUBROUTINE AND THEN CLEARS JUST ONE OF THEM.  THE "CMPREG"
2059                                    ;*      SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
2060                                    ;*      THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
2061                                    ;*      PAR OR PDR ADDRESS.  THE "CMPREG" SUBROUTINE REPORTS THE
2062                                    ;*      ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
2063                                    ;*      ANOTHER REGISTER WAS CLEARED
2064                                    ;*
2065                                    ;*
2066                                    ;*      THE PAR AND PDR CHIPS, PAR/PDR ADDR  MUX, AND ADDR  DETECTION
2067                                    ;*      LOGIC ARE CHECKED
2068                                    ;*
2069                                    ;;*****************************************************************
2070  022742  000004                    TST21   SCOPE
2071
2072  022744  012737  022762  001110    1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER 2$
2073  022752  012703  000010                    MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
2074  022756  012700  172300                    MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2075  022762  012706  001100            2$      MOV     #KERSTK,KSP     ;SETUP STACK POINTER
2076  022766  004737  036070                    JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S IN PDR'S
2077  022772  005010                            CLR     (R0)            ;CLEAR ONE OF THE KERNEL PDR'S
2078  022774  004737  036162                    JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2079  023000  062700  000002                    ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
2080  023004  077312                            SOB     R3,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2081  023006  012737  023024  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2082  023014  012703  000010                    MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
2083  023020  012700  172340                    MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2084  023024  012706  001100            3$      MOV     #KERSTK,KSP     ;SETUP STACK POINTER
2085  023030  004737  036070                    JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
2086  023034  005010                            CLR     (R0)            ;CLEAR ONE OF THE KERNEL PAR'S
2087  023036  004737  036162                    JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2088  023042  062700  000002                    ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
2089  023046  077312                            SOB     R3,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
2090  023050  012737  023066  001110            MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2091  023056  012703  000010                    MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
2092  023062  012700  177600                    MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2093  023066  012706  001100            4$      MOV     #KERSTK,KSP     ;SETUP STACK POINTER
2094  023072  004737  036070                    JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
2095  023076  005010                            CLR     (R0)            ;CLEAR ONE OF THE USER PDR'S
2096  023100  004737  036162                    JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2097  023104  062700  000002                    ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PDR TO CLEAR
2098  023110  077312                            SOB     R3,4$           ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
2099  023112  012737  023130  001110            MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2100  023120  012703  000010                    MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
2101  023124  012700  177640                    MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2102  023130  012706  001100            5$      MOV     #KERSTK,KSP     ;SETUP STACK POINTER
2103  023134  004737  036070                    JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
2104  023140  005010                            CLR     (R0)            ;CLEAR ONE OF THE USER PAR'S
2105  023142  004737  036162                    JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
```

```
2106  023146  062700  000002              ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PAR TO CLEAR
2107  023152  077312                      SOB     R3,5$           ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
2108  023154  012737  022744  001110      MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2109
2110                                  ;;**********************************************************************
2111                                  ;*TEST 22        TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
2112                                  ;*
2113                                  ;*      THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
2114                                  ;*      NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION   THE
2115                                  ;*      "SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
2116                                  ;*      THE PAR/PDR'S   THEN THEY ARE READ TO SEE THAT THEY REMAINED
2117                                  ;*      UNCHANGED
2118                                  ;*
2119                                  ;;**********************************************************************
2120  023162  000004                  TST22   SCOPE
2121
2122
2123  023164  004737  036070          1$      JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
2124  023170  000005                          RESET                   ;ISSUE AN "INIT" BY EXECUTING A RESET
2125  023172  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2126  023176  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2127  023202  011001                  2$      MOV     (R0),R1         ;READ A KERNEL PDR INTO R1
2128  023204  022701  077416                  CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
2129  023210  001401                          BEQ     3$              ;BRANCH IF YES
2130  023212  104055                          ERROR   55              ;KERNEL PDR AFFECTED BY A RESET
2131                                                                  ;FOR TIGHTER SCOPE LOOP
2132                                                                  ;REPLACE ERROR CALL WITH
2133                                                                  ;"BR 2$" = 000773
2134  023214  062700  000002          3$      ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR
2135  023220  077410                          SOB     R4,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR S CHECKED
2136  023222  012700  172340                  MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2137  023226  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2138  023232  011001                  4$      MOV     (R0),R1         ;READ A KERNEL PAR INTO R1
2139  023234  022701  007777                  CMP     #7777,R1        ;ARE ALL THE BITS STILL SET?
2140  023240  001401                          BEQ     5$              ;BRANCH IF YES
2141  023242  104055                          ERROR   55              ;KERNEL PAR AFFECTED BY A RESET
2142                                                                  ;FOR TIGHTER SCOPE LOOP
2143                                                                  ;REPLACE ERROR CALL WITH
2144                                                                  ;"BR 4$" = 000773
2145  023244  062700  000002          5$      ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR
2146  023250  077410                          SOB     R4,4$           ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2147  023252  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2148  023256  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2149  023262  011001                  6$      MOV     (R0),R1         ;READ A USER PDR INTO R1
2150  023264  022701  077416                  CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
2151  023270  001401                          BEQ     7$              ;BRANCH IF YES
2152  023272  104055                          ERROR   55              ;USER PDR AFFECTED BY A RESET
2153                                                                  ;FOR TIGHTER SCOPE LOOP
2154                                                                  ;REPLACE ERROR CALL WITH
2155                                                                  ;"BR 6$" = 000773
2156  023274  062700  000002          7$      ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PDR
2157  023300  077410                          SOB     R4,6$           ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
2158
2159  023302  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2160  023306  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2161  023312  011001                  8$      MOV     (R0),R1         ;READ A USER PAR INTO R1
```

```
2162  023314  022701  007777              CMP     #7777,R1        ;ARE ALL THE BITS STILL SET?
2163  023320  001401                      BEQ     9$              ;BRANCH IF YES
2164  023322  104055                      ERROR   55              ;USER PAR AFFECTED BY A RESET
2165                                                              ;FOR TIGHTER SCOPE LOOP
2166                                                              ;REPLACE ERROR CALL WITH
2167                                                              ;"BR 8$" = 000773
2168  023324  062700  000002      9$      ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PAR
2169  023330  077410                      SOB     R4,8$           ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182                                  ;;*****************************************************************
2183                                  ;*TEST 23       INSTRUCTION FETCH NOT RELOCATED IN MAINT  MODE
2184                                  ;*
2185                                  ;*      THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
2186                                  ;*      MAINTENANCE MODE (DESTINATION-ONLY-RELOCATION), AN INSTRUCTION
2187                                  ;*      FETCH IS NOT RELOCATED AND A RESET CLEARS THE MAINTENANCE BIT
2188                                  ;*      (BIT 08) IN SR0   IF THE "FETCH" IS RELOCATED, A PG.LENGTH ABORT
2189                                  ;*      SHOULD OCCUR, CAUSING A HALT SINCE TRAP CATCHER IS PLACED IN VECTOR 250
2190                                  ;*
2191                                  ;*
2192                                  ;*      NOTE    A HALT MAY OCCUR IF MAINT  MODE NOT DISABLED BY RESET
2193                                  ;*
2194                                  ;;*****************************************************************
2195  023332  000004                  TST23   SCOPE
2196  023334  004737  036002          1$      JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2197  023340  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
2198  023344  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2199  023350  005020                  2$      CLR     (R0)+           ;CLEAR PDR - MAPPING PAGE NON-RES, 0 BLKS
2200  023352  077402                          SOB     R4,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
2201  023354  012737  000252  000250          MOV     #MMVEC+2,MMVEC  ;LOAD TRAP CATCHER INTO MEM MGM. VECTOR
2202  023362  005037  000252                  CLR     MMVEC+2         ;
2203                                                                  ;A HALT WILL OCCUR IF RESET FAILS
2204                                                                  ;TO DISABLE DEST -ONLY RELOCATION
2205  023366  012737  001006  172300          MOV     #1006,KIPDR0    ;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG
2206  023374  012737  023402  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2207  023402  012737  000400  177572  3$      MOV     #BIT8,SR0       ;TURN ON DEST-ONLY-RELOCATION
2208  023410  000005                          RESET                   ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
2209  023412  032737  000400  177572          BIT     #BIT8,SR0       ;WAS MAINT. BIT (BIT 8) OF SR0 CLEARED?
2210  023420  001403                          BEQ     4$              ;BRANCH IF YES
2211  023422  005037  177572                  CLR     SR0             ;CLEAR SR0 SO ERROR CAN BE REPORTED
2212  023426  104061                          ERROR   61              ;MAINT. MODE NOT DISABLED BY A RESET
2213                                                                  ;FOR A TIGHTER SCOPE LOOP
2214                                                                  ;REPLACE ERROR CALL WITH
2215                                                                  ;"BR 3$" = 000765
2216  023430  012706  001100          4$      MOV     #KERSTK,KSP     ;RESTORE STACK POINTER
2217  023434  005037  177572                  CLR     SR0             ;BE SURE SR0 IS CLEAR
```

```
2218  023440  012737  002400  000250          MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT  TRAP VECTOR
2219  023446  012737  000340  000252          MOV     #340,MMVEC+2    ;RESTORE MEM. MGMT VECTOR+2
2220  023454  012737  023334  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2221  023462  004737  036036                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235                                  ;*****************************************************************
2236                                  ;*TEST 24       TEST THAT SOURCE NOT RELOCATED IN MAINTENANCE MODE
2237                                  ;*
2238                                  ;*      THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
2239                                  ;*      MAINTENANCE MODE, THE SOURCE IS NOT RELOCATED.  ONLY THE
2240                                  ;*      DESTINATION IS RELOCATED.  KERNEL PAR'S 3 & 4 ARE MAPPED TO
2241                                  ;*      PHYSICAL ADDRESS 60000-77776.  PDR4 IS SET TO ALLOW FULL READ/WRITE
2242                                  ;*      BUT PDR3 IS CLEAR ALLOWING TO ACCESS.  VIRTUAL ADDRESSES REFERENCING
2243                                  ;*      PAR/PDR'S 4 & 3 ARE USED IN AS DESTINATION AND SOURCE RESPECTIVELY
2244                                  ;*      IF THE SOURCE IS RELOCATED IN MAINTENANCE A MEM. MGMT. TRAP WILL
2245                                  ;*      OCCUR AND THE ERROR WILL BE REPORTED.  KERNEL PG. 7 IS MAPPED R/W
2246                                  ;*
2247                                  ;*****************************************************************
2248  023466  000004                  TST24   SCOPE
2249  023470  004737  036002          1$      JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2250  023474  012737  023564  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2251  023502  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNAL PAGE 3 TO 12-16K
2252  023510  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNAL PAGE 4 TO 12-16K
2253  023516  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNAL PAGE 7 TO THE I/O PAGE
2254  023524  005037  172306                  CLR     KIPDR3          ;MAP KERNAL PAGE 3 "NO ACCESS"
2256  023530  012737  077406  172310          MOV     #77406,KIPDR4   ;MAP KERNAL PAGE 4 R/W, 128 BLKS
2256  023536  012737  077406  172316          MOV     #77406,KIPDR7   ;MAP KERNAL PAGE 7 R/W, 128 BLKS
2257  023544  012737  023644  000250          MOV     #4$,MMVEC       ;SET M.M. VECTOR TO 4$ IN CASE OF ABORT
2258  023552  012737  000377  060000          MOV     #377,@#60000    ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC
2259  023560  012700  177777                  MOV     #-1,R0          ;LOAD EXPECTED CONTENTS OF TEST LOC  IN R0
2260  023564  062737  000400  177572  2$      BIS     #BIT8,SR0       ;TURN ON DEST -ONLY-RELOCATION
2261  023572  113737  060000  100001          MOVB    @#60000,@#100001;LOAD HI BYTE OF TEST LOC -ABORT IF SOURCE RELOCATED
2262  023600  000006                          RESET                   ;TURN OFF DEST. -ONLY-RELOCATION
2263  023602  013701  060000                  MOV     @#60000,R1      ;READ CONTENTS OF TEST LOC
2264  023606  020001                          CMP     R0,R1           ;WAS TEST LOCATION LOADED PROPERLY?
2266  023610  001401                          BEQ     3$              ;BRANCH IF YES
2266  023612  104062                          ERROR   62              ;TEST LOC  NOT LOADED - INST  WAS ABORTED
2267                                                                  ;WHEN SOURCE WAS RELOCATED
2268                                                                  ;FOR TIGHTER SCOPE LOOP REPLACE
2269                                                                  ;ERROR CALL WITH
2270                                                                  ;"BR 2$" = 000764
2271  023614  012737  002400  000250  3$      MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT  VECTOR
2272  023622  012737  077406  172306          MOV     #77406,KIPDR3   ;MAP KERNAL PAGE 3 R/W, 128 BLKS
2273  023630  012737  023470  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
```

```
2274   023636  004737  036036                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2275   023642  000430                          BR      TST25           ;;SKIP TO NEXT TEST
2276                                   ;*  IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 4$
2277
2278   023644  042737  000400  177572  4$      BIC     #BIT8,SR0       ;TURN OFF DEST -ONLY-RELOCATION THRU PAR/PDR 7
2279   023652  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE REST OF SR0 CONTENTS
2280   023660  013737  177576  001370          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2
2281   023666  010637  001354                  MOV     SP,WASR6        ;SAVE VALUE OF THE STACK POINTER
2282   023672  012637  001356                  MOV     (SP)+,TRAPPC    ;SAVE PC OF TRAP
2283   023676  012637  001360                  MOV     (SP)+,TRAPPS    ;SAVE PSW OF TRAP
2284   023702  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR1
2285   023710  104063                          ERROR   63              ;SOURCE APPARENTLY RELOCATED IN MAINT  MODE
2286                                                                   ;FOR TIGHTER SCOPE LOOP
2287                                                                   ;REPLACE ERROR CALL WITH A
2288                                                                   ;"NOP" = 000240
2289   023712  013746  001360                  MOV     TRAPPS,-(SP)    ;PUT PSW OF TRAP BACK ON THE STACK
2290   023716  013746  001356                  MOV     TRAPPC,-(SP)    ;PUT PC OF TRAP BACK ON THE STACK
2291   023722  000002                          RTI                     ;RETURN TO TEST
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305                                   ;;*************************************************************
2306                                   ;*TEST 25       RELOCATION & ADDER TEST (NO CARRIES)
2307                                   ;*
2308                                   ;*      THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
2309                                   ;*      FOR THE REST OF THE PROGRAM.  IT THEN USES DIFFERENT
2310                                   ;*      VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
2311                                   ;*      TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE THREE
2312                                   ;*      MEMORY MANAGEMENT ADDER CHIPS.  THE VALUES ARE SUCH
2313                                   ;*      THAT NO CARRIES ARE GENERATED OUT OF ANY OF THE ADDER CHIPS
2314                                   ;*
2315                                   ;*      THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
2316                                   ;*      IS FORMED BY THE ADDERS IS TO WRITE A PATTERN TO VIRTUAL
2317                                   ;*      LOCATION WITH MEMORY MGMT. IN THE MAINTENANCE MODE, AND
2318                                   ;*      THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
2319                                   ;*      HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR
2320                                   ;*
2321                                   ;;*************************************************************
2322   023724  000004                  TST25   SCOPE
2323
2324   023726  012700  172340          1$      MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2325   023732  005001                          CLR     R1              ;CLEAR R1
2326   023734  012702  000007                  MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
2327   023740  010120                  2$      MOV     R1,(R0)+        ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
2328   023742  062701  000200                  ADD     #200,R1
2329   023746  077204                          SOB     R2,2$           ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
```

```
2330  023750  012710  007600                    MOV     #7600,(R0)      ;MAP KIPAR7 TO THE I/O PAGE
2331  023754  012700  172300                    MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2332  023760  012701  077406                    MOV     #77406,R1       ;LOAD PDR DATA INTO R1
2333  023764  012702  000010                    MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
2334  023770  010120                    3$      MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2335  023772  077202                            SOB     R2,3$                   EXPANDABLE, READ/WRITE
2336
2337  023774  012737  023774  001110  4$      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2338  024002  012700  067776                    MOV     #67776,R0       ;LOAD PHYSICAL ADDR  PBA INTO R0
2339  024006  012701  107776                    MOV     #107776,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2340  024012  012702  125250                    MOV     #125250,R2      ;LOAD TEST PATTERN INTO R2
2341  024016  012704  000600                    MOV     #600,R4 ;LOAD R4 WITH PAR VALUE
2342  024022  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS (11 00)
2343  024026  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2344  024032  052737  000400  177572            BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2345  024040  010211                            MOV     R2,(R1)         ;LOAD 125250 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2346  024042  011003                            MOV     (R0),R3         ;READ 125250 BACK WITHOUT USING MEM. MGMT
2347  024044  000005                            RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2348  024046  013710  001176                    MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC
2349  024052  020203                            CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2350                                                                    ;WRITTEN USING "DEST-ONLY-RELOC."?
2351  024054  001405                            BEQ     5$              ;BRANCH IF YES
2352  024056  010137  001402                    MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS  ADDR
2353  024062  004737  036354                    JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2354  024066  104017                            ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2355                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT
2356                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2357                                                                    ;FORMED WRONG BY ADDERS USING
2358                                                                    ;THE VIRTUAL ADDR. AND KIPAR4
2359                                                                    ;FOR TIGHTER SCOPE LOOP
2360                                                                    ;REPLACE ERROR CALL WITH
2361                                                                    ;"BR 4$" = 000742
2362  024070                            5$
2363  024070  012737  024070  001110  6$      MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2364  024076  012700  067776                    MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2365  024102  012701  102576                    MOV     #102576,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2366  024106  012702  125251                    MOV     #125251,R2      ;LOAD TEST PATTERN INTO R2
2367  024112  012704  000652                    MOV     #652,R4 ;LOAD R4 WITH PAR VALUE
2368  024116  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS (11.00)
2369  024122  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2370  024126  052737  000400  177572            BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2371  024134  010211                            MOV     R2,(R1)         ;LOAD 125251 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2372  024136  011003                            MOV     (R0),R3         ;READ 125251 BACK WITHOUT USING MEM. MGMT
2373  024140  000005                            RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2374  024142  013710  001176                    MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC
2375  024146  020203                            CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2376                                                                    ;WRITTEN USING "DEST-ONLY-RELOC "?
2377  024150  001405                            BEQ     7$              ;BRANCH IF YES
2378  024152  010137  001402                    MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2379  024156  004737  036354                    JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2380  024162  104017                            ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2381                                                                    ;THAT SHOULD HAVE BEEN WRITTEN TO IT
2382                                                                    ;APPARENTLY PHYSICAL ADDR. WAS
2383                                                                    ;FORMED WRONG BY ADDERS USING
2384                                                                    ;THE VIRTUAL ADDR  AND KIPAR4
2385                                                                    ;FOR TIGHTER SCOPE LOOP
```

```
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 6$" = 000742
                                                7$
024164  012737  024164  001110  8$      MOV     #8$,$LPERR            ;SET LOOP ON ERROR POINTER TO 8$
024172  012700  067776                  MOV     #67776,R0             ;LOAD PHYSICAL ADDR. PBA INTO R0
024176  012701  105276                  MOV     #105276,R1            ;LOAD VIRTUAL ADDR. VBA INTO R1
024202  012702  125252                  MOV     #125252,R2            ;LOAD TEST PATTERN INTO R2
024206  012704  000625                  MOV     #625,R4 ;LOAD R4 WITH PAR VALUE
024212  010437  172350                  MOV     R4,KIPAR4             ;LOAD KERNEL PAR 4 BITS (11:00)
024216  011037  001176                  MOV     (R0),$TMP0            ;SAVE CONTENTS AT TEST LOCATION
024222  052737  000400  177572          BIS     #BIT8,SR0             ;TURN ON "DESTINATION-ONLY-RELOCATION"
024230  010211                          MOV     R2,(R1)               ;LOAD 125252 USING ADDER CHIPS (PAR4 + VIRT ADDR )
024232  011003                          MOV     (R0),R3               ;READ 125252 BACK WITHOUT USING MEM. MGMT
024234  000005                          RESET                         ;TURN OFF MEMORY MGMT. MAINT. MODE
024236  013710  001176                  MOV     $TMP0,(R0)            ;RESTORE ORIGINAL CONTENTS TO TEST LOC
024242  020203                          CMP     R2,R3                 ;WAS SAME PATTERN READ BACK THAT WAS
                                                                        ;WRITTEN USING "DEST-ONLY-RELOC."?
024244  001405                          BEQ     9$                    ;BRANCH IF YES
024246  010137  001402                  MOV     R1,VIRT1              ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
024252  004737  036354                  JSR     PC,FORMPA             ;GO FORM PHYSICAL ADDRESS FOR TYPING
024256  104017                          ERROR   17                    ;TEST LOCATION DID NOT HAVE PATTERN
                                                                        ;THAT SHOULD HAVE BEEN WRITTEN TO IT
                                                                        ;APPARENTLY PHYSICAL ADDR. WAS
                                                                        ;FORMED WRONG BY ADDERS USING
                                                                        ;THE VIRTUAL ADDR. AND KIPAR4
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 8$" = 000742
024260                                  9$
                                                                        ;
024260  012737  024260  001110  10$     MOV     #10$,$LPERR           ;SET LOOP ON ERROR POINTER TO 10$
024266  012700  177776                  MOV     #PSW,R0               ;LOAD PHYS. ADDR. OF PSW INTO R0
024272  012701  100076                  MOV     #100076,R1            ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
024276  012702  030340                  MOV     #030340,R2            ;LOAD DATA FOR PSW IN R2
024302  012704  007777                  MOV     #7777,R4              ;LOAD R4 WITH PAR VALUE
024306  010437  172350                  MOV     R4,KIPAR4             ;LOAD KERNEL PAR 4 BITS (11:00)
024312  005010                          CLR     (R0)                  ;CLEAR THE PSW
024314  052737  000400  177572          BIS     #BIT8,SR0             ;TURN ON "DESTINATION-ONLY-RELOCATION"
024322  010211                          MOV     R2,(R1)               ;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT ADDR )
024324  011003                          MOV     (R0),R3               ;READ PSW BACK WITHOUT USING MEM. MGMT
024326  000005                          RESET                         ;TURN OFF MEM. MGMT MAINT. MODE
024330  005010                          CLR     (R0)                  ;CLEAR THE PSW
024332  042703  000037                  BIC     #37,R3                ;MASK T-BIT & CC BITS OUT OF DATA READ
024336  020203                          CMP     R2,R3                 ;WAS PSW WRITTEN WHILE IN MAINT. MODE?
024340  001406                          BEQ     11$                   ;BRANCH IF YES
024342  010137  001402                  MOV     R1,VIRT1              ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
024346  004737  036354                  JSR     PC,FORMPA             ;GO FORM PHYSICAL ADDR. FOR TYPING
024352  104017                          ERROR   17                    ;PSW DID NOT HAVE DATA THAT IT SHOULD HAVE.
                                                                        ;APPARENTLY PHYS. ADDR. OF PSW WAS
                                                                        ;NOT FORMED BY ADDERS USING THE
                                                                        ;VIRTUAL ADDR. AND KIPAR4
                                                                        ;FOR TIGHTER SCOPE LOOP
                                                                        ;REPLACE ERROR CALL WITH
                                                                        ;"BR 10$" = 000742
024354  012737  024354  001110  11$     MOV     #11$,$LPERR           ;SET LOOP ON ERROR POINTER TO 11$
024362  012700  177776                  MOV     #PSW,R0               ;LOAD PHYS. ADDR. OF PSW INTO R0
```

```
2442  024366  012701  117776          MOV     #117776,R1      ;LOAD VIRTUAL ADDR  FOR PSW INTO R1
2443  024372  012702  030240          MOV     #030240,R2      ;LOAD DATA FOR PSW IN R2
2444  024376  012704  007600          MOV     #7600,R4        ;LOAD R4 WITH PAR VALUE
2445  024402  010437  172350          MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS (11 00)
2446  024406  052737  000400  177572  BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2447  024414  010211                  MOV     R2,(R1)         ;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT  ADDR )
2448  024416  011003                  MOV     (R0),R3         ;READ PSW BACK WITHOUT USING MEM  MGMT
2449  024420  000005                  RESET                   ;TURN OFF MEM  MGMT. MAINT.  MODE
2450  024422  005010                  CLR     (R0)            ;CLEAR THE PSW
2451  024424  042703  000037          BIC     #37,R3          ;MASK T-BIT & CC BITS OUT OF DATA READ
2452  024430  020203                  CMP     R2,R3           ;WAS PSW WRITTEN WHILE IN MAINT.  MODE?
2453  024432  001405                  BEQ     12$             ;BRANCH IF YES
2454  024434  010137  001402          MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR
2455  024440  004737  036354          JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDR. FOR TYPING
2456  024444  104017                  ERROR   17              ;PSW DID NOT HAVE DATA THAT IT SHOULD
2457                                                          ;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2458                                                          ;NOT FORMED BY ADDERS USING THE
2459                                                          ;VIRTUAL ADDR. AND KIPAR4
2460                                                          ;FOR TIGHTER SCOPE LOOP
2461                                                          ;REPLACE ERROR CALL WITH
2462                                                          ;"BR 11$" = 000743
2463  024446  012737  023726  001110  12$     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2464
2465                                  ;;*****************************************************************
2466                                  ;*TEST 26        RELOCATION & ADDER TEST (WITH CARRIES)
2467                                  ;*
2468                                  ;*      THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2469                                  ;*      TEST TO VERIFY MEMORY MANAGEMENTS ABILITY TO CONSTRUCT
2470                                  ;*      PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2471                                  ;*      CONTENTS OF A PAGE ADDRESS REGISTER.  HOWEVER, THE VALUES
2472                                  ;*      AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES FROM
2473                                  ;*      CHIP TO CHIP AND CHECK "WRAPAROUND" TO ADDRESS 000000 BY
2474                                  ;*      USING VIRTUAL ADDR. 100100 AND KIPAR4 = 7777.
2475                                  ;*
2476                                  ;;*****************************************************************
2477  024454  000004                  TST26.  SCOPE
2478
2479  024456                          1$:                             ;KERNEL PAR'S AND PDR'S HAVE BEEN
2480                                                                  ;SETUP BY THE PREVIOUS TEST
2481  024456  012737  024456  001110  2$:      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2482  024464  012700  064276          MOV     #64276,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2483  024470  012701  102176          MOV     #102176,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2484  024474  012702  125253          MOV     #125253,R2      ;LOAD TEST PATTERN INTO R2
2485  024500  012704  000621          MOV     #621,R4 ;LOAD R4 WITH PAR VALUE
2486  024504  010437  172350          MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS (11:00)
2487  024510  011037  001176          MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2488  024514  062737  000400  177572  BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2489  024522  010211                  MOV     R2,(R1)         ;LOAD 125253 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2490  024524  011003                  MOV     (R0),R3         ;READ 125253 BACK WITHOUT USING MEM  MGMT
2491  024526  000006                  RESET                   ;TURN OFF MEMORY MGMT.  MAINT.  MODE
2492  024530  013710  001176          MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC
2493  024534  020203                  CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2494                                                          ;WRITTEN USING "DEST-ONLY-RELOC."?
2495  024536  001405                  BEQ     3$              ;BRANCH IF YES
2496  024540  010137  001402          MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR.  TO FORM PHYS  ADDR
2497  024544  004737  036354          JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
```

```
2498  024550  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2499                                                                  ;THAT SHOULD HAVE PEEN WRITTEN TO IT
2500                                                                  ;APPARENTLY PHYSICAL ADDR  WAS
2501                                                                  ;FORMED WRONG BY ADDERS USING
2502                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2503                                                                  ;FOR TIGHTER SCOPE LOOP
2504                                                                  ;REPLACE ERROR CALL WITH
2505                                                                  ;"BR 2$" = 000742
2506  024552                            3$
2507  024552  012737  024552  001110    4$      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2508  024560  012700  064476                    MOV     #64476,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2509  024564  012701  112376                    MOV     #112376,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2510  024570  012702  125254                    MOV     #125254,R2      ;LOAD TEST PATTERN INTO R2
2511  024574  012704  000521                    MOV     #521,R4 ;LOAD R4 WITH PAR VALUE
2512  024600  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2513  024604  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2514  024610  052737  000400  177572          BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2515  024616  010211                          MOV     R2,(R1)         ;LOAD 125254 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2516  024620  011003                          MOV     (R0),R3         ;READ 125254 BACK WITHOUT USING MEM  MGMT
2517  024622  000005                          RESET                   ;TURN OFF MEMORY MGMT. MAINT. MODE
2518  024624  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC
2519  024630  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2520                                                                  ;WRITTEN USING "DEST-ONLY-RELOC "?
2521  024632  001405                          BEQ     5$              ;BRANCH IF YES
2522  024634  010137  001402                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2523  024640  004737  036354                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2524  024644  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2525                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT
2526                                                                  ;APPARENTLY PHYSICAL ADDR  WAS
2527                                                                  ;FORMED WRONG BY ADDERS USING
2528                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2529                                                                  ;FOR TIGHTER SCOPE LOOP
2530                                                                  ;REPLACE ERROR CALL WITH
2531                                                                  ;"BR 4$" = 000742
2532  024646                            5$
2533  024646  012737  024646  001110    6$      MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2534  024664  012700  061076                    MOV     #61076,R0       ;LOAD PHYSICAL ADDR  PBA INTO R0
2535  024660  012701  106776                    MOV     #106776,R1      ;LOAD VIRTUAL ADDR  VBA INTO R1
2536  024664  012702  125255                    MOV     #125255,R2      ;LOAD TEST PATTERN INTO R2
2537  024670  012704  000521                    MOV     #521,R4 ;LOAD R4 WITH PAR VALUE
2538  024674  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11 00>
2539  024700  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2540  024704  052737  000400  177572          BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2541  024712  010211                          MOV     R2,(R1)         ;LOAD 125255 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2542  024714  011003                          MOV     (R0),R3         ;READ 125255 BACK WITHOUT USING MEM  MGMT
2543  024716  000005                          RESET                   ;TURN OFF MEMORY MGMT  MAINT. MODE
2544  024720  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC
2545  024724  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2546                                                                  ;WRITTEN USING "DEST-ONLY-RELOC "?
2547  024726  001405                          BEQ     7$              ;BRANCH IF YES
2548  024730  010137  001402                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS  ADDR
2549  024734  004737  036354                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2560  024740  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2561                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT
2562                                                                  ;APPARENTLY PHYSICAL ADDR  WAS
2563                                                                  ;FORMED WRONG BY ADDERS USING
```

```
2554                                                                    ,THE VIRTUAL ADDR. AND KIPAR4
2555                                                                    ,FOR TIGHTER SCOPE LOOP
2556                                                                    ,REPLACE ERROR CALL WITH
2557                                                                    ,"BR 6$" = 000742
2558  024742                                    7$
2559  024742  012737  024742  001110  8$        MOV     #8$,$LPERR      ,SET LOOP ON ERROR POINTER TO 8$
2560  024750  012700  060076                    MOV     #60076,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2561  024754  012701  107776                    MOV     #107776,R1      ;LOAD VIRTUAL ADDR  VBA INTO R1
2562  024760  012702  125256                    MOV     #125256,R2      ;LOAD TEST PATTERN INTO R2
2563  024764  012704  000501                    MOV     #501,R4 ,LOAD R4 WITH PAR VALUE
2564  024770  010437  172350                    MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <11:00>
2565  024774  011037  001176                    MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2566  025000  052737  000400  177572            BIS     #BIT8,SR0       ,TURN ON "DESTINATION-ONLY-RELOCATION"
2567  025006  010211                            MOV     R2,(R1)         ,LOAD 125256 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2568  025010  011003                            MOV     (R0),R3         ,READ 125256 BACK WITHOUT USING MEM  MGMT
2569  025012  000005                            RESET                   ,TURN OFF MEMORY MGMT. MAINT. MODE
2570  025014  013710  001176                    MOV     $TMP0,(R0)      ,RESTORE ORIGINAL CONTENTS TO TEST LOC
2571  025020  020203                            CMP     R2,R3           ,WAS SAME PATTERN READ BACK THAT WAS
2572                                                                    ,WRITTEN USING "DEST-ONLY-RELOC."?
2573  025022  001405                            BEQ     9$              ;BRANCH IF YES
2574  025024  010137  001402                    MOV     R1,VIRT1        ,SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2575  025030  004737  036354                    JSR     PC,FORMPA       ,GO FORM PHYSICAL ADDRESS FOR TYPING
2576  025034  104017                            ERROR   17              ,TEST LOCATION DID NOT HAVE PATTERN
2577                                                                    ,THAT SHOULD HAVE BEEN WRITTEN TO IT
2578                                                                    ,APPARENTLY PHYSICAL ADDR. WAS
2579                                                                    ,FORMED WRONG BY ADDERS USING
2580                                                                    ,THE VIRTUAL ADDR. AND KIPAR4
2581                                                                    ,FOR TIGHTER SCOPE LOOP
2582                                                                    ,REPLACE ERROR CALL WITH
2583                                                                    ,"BR 8$" = 000742
2584  025036                                    9$
2585  025036  012737  025036  001110  10$       MOV     #10$,$LPERR     ,SET LOOP ON ERROR POINTER TO 10$
2586  025044  012700  000000                    MOV     #000000,R0      ,LOAD PHYSICAL ADDR. PBA INTO R0
2587  025050  012701  100100                    MOV     #100100,R1      ,LOAD VIRTUAL ADDR. VBA INTO R1
2588  025054  012702  125257                    MOV     #125257,R2      ,LOAD TEST PATTERN INTO R2
2589  025060  012704  007777                    MOV     #7777,R4        ,LOAD R4 WITH PAR VALUE
2590  025064  010437  172350                    MOV     R4,KIPAR4       ,LOAD KERNEL PAR 4 BITS <11 00>
2591  025070  011037  001176                    MOV     (R0),$TMP0      ,SAVE CONTENTS AT TEST LOCATION
2592  025074  052737  000400  177572            BIS     #BIT8,SR0       ;TURN ON "DESTINATION-ONLY-RELOCATION"
2593  025102  010211                            MOV     R2,(R1)         ,LOAD 125257 USING ADDER CHIPS (PAR4 + VIRT ADDR )
2594  025104  011003                            MOV     (R0),R3         ,READ 125257 BACK WITHOUT USING MEM  MGMT
2595  025106  000005                            RESET                   ,TURN OFF MEMORY MGMT. MAINT. MODE
2596  025110  013710  001176                    MOV     $TMP0,(R0)      ,RESTORE ORIGINAL CONTENTS TO TEST LOC
2597  025114  020203                            CMP     R2,R3           ,WAS SAME PATTERN READ BACK THAT WAS
2598                                                                    ,WRITTEN USING "DEST-ONLY-RELOC "?
2599  025116  001405                            BEQ     11$             ,BRANCH IF YES
2600  025120  010137  001402                    MOV     R1,VIRT1        ,SAVE VIRTUAL ADDR  TO FORM PHYS  ADDR
2601  025124  004737  036354                    JSR     PC,FORMPA       ,GO FORM PHYSICAL ADDRESS FOR TYPING
2602  025130  104017                            ERROR   17              ,TEST LOCATION DID NOT HAVE PATTERN
2603                                                                    ,THAT SHOULD HAVE BEEN WRITTEN TO IT
2604                                                                    ,APPARENTLY PHYSICAL ADDR  WAS
2605                                                                    ,FORMED WRONG BY ADDERS USING
2606                                                                    ,THE VIRTUAL ADDR. AND KIPAR4
2607                                                                    ,FOR TIGHTER SCOPE LOOP
2608                                                                    ,REPLACE ERROR CALL WITH
2609                                                                    ,"BR 10$" = 000742
```

```
2610  025132                          11$
2611  025132  012737  024456  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2612
2613                                  ;;*****************************************************************
2614                                  ;*TEST 27       READ AND WRITE WHILE IN RELOCATE MODE
2615                                  ;*
2616                                  ;*      THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2617                                  ;*      READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2618                                  ;*      060000-067600.  ONE LOCATION IN EVERY BLOCK (32. WORDS)
2619                                  ;*      IS WRITTEN USING PAR4 AND READ USING PAR5.  THIS IS
2620                                  ;*      DONE IN BOTH USER AND KERNEL MODES.  THE USER PAR/PDR'S
2621                                  ;*      ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2622                                  ;*      MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2623                                  ;*      OF THE PROGRAM.  THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
2624                                  ;*      IS CHECKED BY READING AND WRITING IN USER MODE.  REMEMBER
2625                                  ;*      ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2626                                  ;*      MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2627                                  ;*      THE PAR/PDR'S TO EXECUTE.
2628                                  ;*
2629                                  ;*      WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
2630                                  ;*      NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2631                                  ;*      PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
2632                                  ;*      USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2633                                  ;*      USER MODE).  IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2634                                  ;*      TO 8$ WHERE THE TRAP IS REPORTED
2635                                  ;*
2636                                  ;;*****************************************************************
2637  025140  000004                  TST27   SCOPE
2638
2639  025142  005037  177776          1$      CLR     PSW             ;START IN KERNEL MODE
2640  025146  012704  000577                  MOV     #577,R4         ;LOAD R4 WITH VALUE FOR PAR4
2641  025152  012705  000600                  MOV     #600,R5         ;LOAD R5 WITH VALUE FOR PAR5
2642  025156  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR4
2643  025162  010537  172352                  MOV     R5,KIPAR5       ;LOAD KERNEL PAR5
2644  025166  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2645  025172  005001                          CLR     R1              ;CLEAR R1
2646  025174  012702  000007                  MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
2647  025200  010120                  2$      MOV     R1,(R0)+        ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2648  025202  062701  000200                  ADD     #200,R1
2649  025206  077204                          SOB     R2,2$           ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2650  025210  012710  007600                  MOV     #7600,(R0)      ;MAP USER PAR7 TO THE I/O PAGE
2651  025214  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2652  025220  012701  077406                  MOV     #77406,R1       ;LOAD PDR DATA INTO R1
2653  025224  012702  000010                  MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
2654  025230  010120                  3$      MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2655  025232  077202                          SOB     R2,3$           ;       EXPANDABLE, READ/WRITE
2656  025234  106037  177610                  CLRB    UIPDR4          ;MAP USER SPACE NON-RESIDENT WHILE
2657  025240  106037  177612                  CLRB    UIPDR5          ;       TESTING KERNEL SPACE
2658  025244  010537  177650                  MOV     R5,UIPAR4       ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2659  025250  010437  177652                  MOV     R4,UIPAR5
2660  025254  012737  000001  177572          MOV     #1,SR0          ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
2661  025262  012737  025314  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2662  025270  012737  025526  000250          MOV     #8$,MMVEC       ;SET M  M  TRAP VECTOR TO 8$
2663  025276  013737  177776  001176  4$      MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
2664  025304  012700  100100                  MOV     #100100,R0      ;PUT VIRTUAL ADDR  THAT USER PAR4 IN R0
2665  025310  012701  120000                  MOV     #120000,R1      ;PUT VIRTUAL ADDR  THAT USES PAR5 IN R1
```

```
2666  025314  010010                  5$      MOV     R0,(R0)         ;WRITE TO TEST LOC  USING PAR4
2667  025316  011102                          MOV     (R1),R2         ;READ THE SAME LOC , BUT USING PAR5
2668  025320  020002                          CMP     R0,R2           ;DID WE READ WHAT WE WROTE?
2669  025322  001411                          BEQ     6$              ;BRANCH IF YES
2670  025324  010137  001404                  MOV     R1,VIRT2        ;SAVE VIRTUAL ADDR.  THAT SELECTED PAR5
2671  025330  010037  001402                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDR.  THAT SELECTED PAR4
2672  025334  004737  036354                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS  BEING USED
2673  025340  104020                          ERROR   20              ;READING LOC. USING PAR5 AND A VIRT
2674                                                                  ;ADDR  DID NOT FIND DATA WRITTEN WHEN USING
2675                                                                  ;PAR4 AND VIRT. ADDRESS
2676                                                                  ;FOR TIGHTER SCOPE LOOP
2677                                                                  ;REPLACE ERROR CALL WITH
2678                                                                  ;"BR 5$" = 000765
2679  025342  013700  001402                          MOV     VIRT1,R0        ;RESTORE VBA IN R0
2680  025346  062700  000100          6$      ADD     #100,R0         ;CHANGE VIRTUAL ADDRS  TO POINT TO NEXT BLOCK
2681  025352  062701  000100                  ADD     #100,R1
2682  025356  020127  127700                  CMP     R1,#127700      ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
2683  025362  001354                          BNE     5$              ;BRANCH IF NO
2684  025364  032737  140000  177776          BIT     #140000,PSW     ;HAVE WE DONE TEST IN USER MODE YET?
2685  025372  001026                          BNE     7$              ;BRANCH IF YES
2686  025374  010437  177650                  MOV     R4,UIPAR4       ;LOAD USER PAR4
2687  025400  010537  177652                  MOV     R5,UIPAR5       ;LOAD USER PAR5
2688  025404  112737  000006  177610          MOVB    #6,UIPDR4       ;MAP USER SPACE R/W TO TEST IT
2689  025412  112737  000006  177612          MOVB    #6,UIPDR5
2690  025420  105037  172310                  CLRB    KIPDR4          ;MAP  KERNEL SPACE NON-RESIDENT WHILE
2691  025424  105037  172312                  CLRB    KIPDR5          ;       TESTING USER SPACE
2692  025430  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2693  025434  010437  172352                  MOV     R4,KIPAR5
2694  025440  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
2695  025446  000713                          BR      4$              ;GO BACK AND READ/WRITE IN USER MODE
2696  025450  005037  177776          7$      CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
2697  025454  012737  077406  172310          MOV     #77406,KIPDR4   ;REMAP KERNEL PAGES READ/WRITE
2698  025462  012737  077406  172312          MOV     #77406,KIPDR5
2699  025470  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL  AND USER PAR'S 4 & 5
2700  025474  010537  172352                  MOV     R5,KIPAR5       ;  BACK TO 12-16K
2701  025600  010537  177650                  MOV     R5,UIPAR4
2702  025604  010537  177652                  MOV     R5,UIPAR5
2703  025610  012737  002400  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDR  OF NORMAL M M  TRAP ROUTINE
2704  025616  012737  025142  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2705  025624  000427                          BR      TST30           ;,GO TO NEXT TEST
2706
2707  025626  012637  001356          8$      MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
2708  025632  012637  001360                  MOV     (KSP)+,TRAPPS
2709                                                                  ;PROGRAM WILL TRAP TO HERE IF TRY
2710                                                                  ;TO USE USER PDR'S WHEN IN KERNEL MODE
2711                                                                  ;OR KERNEL PDR'S WHEN IN USER MODE
2712  025636  010037  001402                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2713  025642  004737  036354                  JSR     PC,FORMPA       ;GO FORM THE PHYSICAL ADDRESS  BEING USED
2714  025646  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE SR0 & SR2  FOR ERROR REPORT
2715  025654  013737  177576  001370          MOV     SR2,WASSR2
2716  025662  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2717  025670  104052                          ERROR   52              ;M M  TRAP WHILE IN RELOCATE MODE -
2718                                                                  ;REFERENCED WRONG SET OF PDR'S
2719                                                                  ;FOR TIGHTER SCOPE LOOP
2720                                                                  ;REPLACE ERROR CALL WITH
2721                                                                  ;A "NOP" = 000240
```

```
2722  025572  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
2723  025576  013746  001356                  MOV     TRAPPC,-(KSP)
2724  025602  000002                          RTI                     ;RETURN TO TEST
2725
2726
2727                                  ;;*****************************************************************
2728                                  ;*TEST 30       W-BIT LOGIC TEST, KERNEL PDR'S
2729                                  ;*
2730                                  ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2731                                  ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2732                                  ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2733                                  ;*      77776,77776, & 777776 RESPECTIVELY).
2734                                  ;*      WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
2735                                  ;*      EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS  THE PDR'S
2736                                  ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2737                                  ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2738                                  ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  KERNEL PDR'S 3,4,5,6
2739                                  ;*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
2740                                  ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO  THE
2741                                  ;*      W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
2742                                  ;*      ARE BEING CHECKED
2743                                  ;;*****************************************************************
2744  025604  000004                  TST30   SCOPE
2745  025606                          1$
2746  025606  004737  036002                  JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2747  025612  012702  000004                  MOV     #4,R2           ;SET LOOP COUNTER TO 4
2748  025616  012700  172346                  MOV     #KIPAR3,R0      ;LOAD ADDRESS OF PAR3 INTO R0
2749  025622  012701  000600                  MOV     #600,R1         ;LOAD "12-16K" PAR VALUE INTO R1
2750  025626  010120                  2$      MOV     R1,(R0)+        ;MAP PARS 3-6 TO 12-16K
2751  025630  077202                          SOB     R2,2$           ;LOOP TIL ALL 4 OF THEM LOADED
2752  025632  012705  172300                  MOV     #KIPDR0,R5      ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2753  025636  012704  000010                  MOV     #10,R4          ;SET LOOP COUNTER TO 8
2754  025642  012703  017776                  MOV     #17776,R3       ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2755  025646  012737  025654  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2756  025654  012700  172300          3$      MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2757  025660  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
2758  025664  012701  077406                  MOV     #77406,R1       ;PUT "W-BIT OFF DATA" INTO R1
2759  025670  010120                  4$      MOV     R1,(R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2760  025672  077202                          SOB     R2,4$           ;LOOP UNTIL ALL OF THEM SETUP
2761  025674  011313                          MOV     (R3),(R3)       ;DO "DATO" TO VIRTUAL ADDR -SETTING A W-BIT
2762  025676  031527  000100                  BIT     (R5),#WBIT      ;DID THAT CAUSE W-BIT TO BE SET?
2763  025702  001002                          BNE     5$              ;BRANCH IF YES
2764  025704  104021                          ERROR   21              ;W-BIT DID NOT GET SET IN PDR
2765                                                                  ;FOR TIGHTER SCOPE LOOP
2766                                                                  ;REPLACE ERROR CALL WITH
2767                                                                  ;"BR 3$" = 000763
2768  025706  000422                          BR      8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2769  025710  012702  000010          5$      MOV     #10,R2          ;SET LOOP COUNTER TO 8
2770  025714  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2771  025720  031027  000100          6$      BIT     (R0),#WBIT      ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2772  025724  001403                          BEQ     7$              ;BRANCH IF YES
2773  025726  020500                          CMP     R5,R0           ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2774  025730  001401                          BEQ     7$              ;BRANCH IF YES
2775  025732  104022                          ERROR   22              ;W-BIT GOT SET IN MORE THAN ONE PDR
2776                                                                  ;FOR TIGHTER SCOPE LOOP
2777                                                                  ;REPLACE ERROR CALL WITH
```

```
2778                                                    ,"BR 3$" = 000750
2779  025734  062700  000002          7$      ADD     #2,R0           ,POINT R0 TO NEXT PDR TO BE CHECKED
2780  025740  077211                          SOB     R2,6$           ,LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2781  025742  010115                          MOV     R1,(R5)         ,WRITE TO THE PDR TESTED TO CLEAR W-BIT
2782  025744  031527  000100                  BIT     (R5),#WBIT      ,DID WRITING PDR CLEAR THE W-BIT?
2783  025750  001401                          BEQ     8$              ,BRANCH IF YES
2784  025752  104023                          ERROR   23              ,W-BIT DID NOT CLEAR BY WRITING THE PDR
2785                                                                  ,FOR TIGHTER SCOPE LOOP
2786                                                                  ,REPLACE ERROR CALL WITH
2787                                                                  ,"BR 3$" = 000740
2788  025754  062705  000002          8$      ADD     #2,R5           ,POINT R5 TO THE NEXT PDR TO BE TESTED
2789  025760  062703  020000                  ADD     #20000,R3       ,CHANGE VIRT. ADDR TO REF  NEXT PDR
2790  025764  077445                          SOB     R4,3$           ,LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2791  025766  012737  025606  001110          MOV     #1$,$LPERR      ,RESET LOOP ON ERROR POINTER TO 1$
2792  025774  004737  036036                  JSR     PC,TON          ,TURN T-BIT BACK ON FOR NEXT TEST
2793
2794                                          ,,*****************************************************************
2795                                          ,*TEST 31       W-BIT LOGIC TEST, USER PDR'S
2796                                          ,*
2797                                          ,*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2798                                          ,*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2799                                          ,*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2800                                          ,*      77776,77776, & 777776 RESPECTIVELY).
2801                                          ,*      WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
2802                                          ,*      EIGHT (8) USER PAGE DESCRIPTOR REGISTERS.  THE PDR'S
2803                                          ,*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2804                                          ,*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2805                                          ,*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  USER PDR'S 3,4,5,6
2806                                          ,*      ARE MAPPED TO 12-16K FOR THIS TEST  ALSO THE W-BIT
2807                                          ,*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO  THE
2808                                          ,*      W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
2809                                          ,*      ARE BEING CHECKED
2810                                          ,,*****************************************************************
2811  026000  000004                  TST31   SCOPE
2812  026002  012737  140000  177776  1$      MOV     #140000,PSW      GO TO USER MODE FOR THIS TEST
2813  026010  004737  036002                  JSR     PC,TOFF         ,TURN T-BIT TRAPPING OFF FOR THIS TEST
2814  026014  012702  000004                  MOV     #4,R2           ,SET LOOP COUNTER TO 4
2815  026020  012700  177646                  MOV     #UIPAR3,R0      ,LOAD ADDRESS OF PAR3 INTO R0
2816  026024  012701  000600                  MOV     #600,R1         ,LOAD "12-16K" PAR VALUE INTO R1
2817  026030  010120                  2$      MOV     R1,(R0)+        ,MAP PARS 3-6 TO 12-16K
2818  026032  077202                          SOB     R2,2$           ,LOOP TIL ALL 4 OF THEM LOADED
2819  026034  012705  177600                  MOV     #UIPDR0,R5      ,LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2820  026040  012704  000010                  MOV     #10,R4          ,SET LOOP COUNTER TO 8
2821  026044  012703  017776                  MOV     #17776,R3       ,INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2822  026050  012737  026056  001110          MOV     #3$,$LPERR      ,SET LOOP ON ERROR POINTER TO 3$
2823  026056  012700  177600          3$      MOV     #UIPDR0,R0      ,LOAD ADDR  OF FIRST PDR TO BE SETUP IN R0
2824  026062  012702  000010                  MOV     #10,R2          ,SET LOOP COUNTER TO 8
2825  026066  012701  077406                  MOV     #77406,R1       ,PUT "W-BIT OFF DATA" INTO R1
2826  026072  010120                  4$      MOV     R1,(R0)+        ,CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2827  026074  077202                          SOB     R2,4$           ,LOOP UNTIL ALL OF THEM SETUP
2828  026076  011313                          MOV     (R3),(R3)       ,DO "DATO" TO VIRTUAL ADDR -SETTING A W-BIT
2829  026100  031527  000100                  BIT     (R5),#WBIT      ,DID THAT CAUSE W-BIT TO BE SET?
2830  026104  001002                          BNE     5$              ,BRANCH IF YES
2831  026106  104021                          ERROR   21              ,W-BIT DID NOT GET SET IN PDR
2832                                                                  ,FOR TIGHTER SCOPE LOOP
2833                                                                  ,REPLACE ERROR CALL WITH
```

```
2834                                                    ,"BR 3$" = 000763
2835  026110  000422                        BR      8$    ,SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2836  026112  012702  000010        5$      MOV     #10,R2    ,SET LOOP COUNTER TO 8
2837  026116  012700  177600                MOV     #UIPDR0,R0    ,LOAD ADDR  OF FIRST PDR TO BE CHECKED IN R0
2838  026122  031027  000100        6$      BIT     (R0),#WBIT    ,DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2839  026126  001403                BEQ     7$    ,BRANCH IF YES
2840  026130  020500                CMP     R5,R0    ,IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2841  026132  001401                BEQ     7$    ,BRANCH IF YES
2842  026134  104022                ERROR   22    ,W-BIT GOT SET IN MORE THAN ONE PDR
2843                                                    ,FOR TIGHTER SCOPE LOOP
2844                                                    ,REPLACE ERROR CALL WITH
2845                                                    ,"BR 3$" = 000750
2846  026136  062700  000002        7$      ADD     #2,R0    ,POINT R0 TO NEXT PDR TO BE CHECKED
2847  026142  077211                SOB     R2,6$    ,LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2848  026144  010115                MOV     R1,(R5)    ,WRITE TO THE PDR TESTED TO CLEAR W-BIT
2849  026146  031527  000100        BIT     (R5),#WBIT    ,DID WRITING PDR CLEAR THE W-BIT?
2850  026152  001401                BEQ     8$    ,BRANCH IF YES
2851  026154  104023                ERROR   23    ,W-BIT DID NOT CLEAR BY WRITING THE PDR
2852                                                    ,FOR TIGHTER SCOPE LOOP
2853                                                    ,REPLACE ERROR CALL WITH
2854                                                    ,"BR 3$" = 000740
2855  026156  062705  000002        8$      ADD     #2,R5    ,POINT R5 TO THE NEXT PDR TO ⌴  TESTED
2856  026162  062703  020000                ADD     #20000,R3    ,CHANGE VIRT. ADDR TO REF. NEXT PDR
2857  026166  077445                SOB     R4,3$    ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2858  026170  012737  026002  001110        MOV     #1$,$LPERR    ,RESET LOOP ON ERROR POINTER TO 1$
2859  026176  004737  036036                JSR     PC,TON    ;TURN T-BIT BACK ON FOR NEXT TEST
2860  026202  005037  177776                CLR     PSW    ;BACK TO KERNEL MODE BEFORE LEAVING
2861
2862                                    ,,**********************************************************************
2863                                    ,*TEST 32       TEST "W-BIT NOT SET" CASES
2864                                    ,*
2865                                    ,*      THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES
2866                                    ,*      NOT GET SET ON A WRITE.  FIRST CASE IS THAT THE W-BIT
2867                                    ,*      SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO
2868                                    ,*      STATUS REG SR0 (KERNEL PDR 7 IS USED).  SECOND CASE IS THAT
2869                                    ,*      THE W-BIT IS NOT SET IF THE "DATO" IS ABORTED DUE TO AN
2870                                    ,*      ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED)
2871                                    ,*
2872                                    ,,**********************************************************************
2873  026206  000004                TST32   SCOPE
2874
2875  026210  004737  036002        1$      JSR     PC,TOFF    ,TURN OFF T-BIT TRAPPING FOR THIS TEST
2876  026214  012701  077406                MOV     #77406,R1    ,PUT "W-BIT OFF" VALUE FOR PDR IN R1
2877  026220  012737  026226  001110        MOV     #2$,$LPERR    ,SET LOOP ON ERROR POINTER TO 2$
2878  026226  010137  172316        2$      MOV     R1,KIPDR7    ,LOAD KERNEL PDR 7 TO CLEAR W-BIT
2879  026232  013700  177572                MOV     SR0,R0    ,READ PRESENT CONTENTS OF STATUS REG  0
2880  026236  010037  177572                MOV     R0,SR0    ,WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
2881  026242  013702  172316                MOV     KIPDR7,R2    ,READ CONTENTS OF KIPDR7 INTO R2
2882  026246  020102                CMP     R1,R2    ,WAS W-BIT LEFT CLEARED?
2883  026250  001401                BEQ     3$    ,BRANCH IF YES
2884  026252  104024                ERROR   24    ,W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
2885                                                    ,FOR TIGHTER SCOPE LOOP
2886                                                    ,REPLACE ERROR CALL WITH
2887                                                    ,"BR 2$" = 000765
2888  026254  012737  026254  001110  3$      MOV     #3$,$LPERR    ,SET LOOP ON ERROR POINTER TO 3$
2889  026262  010137  172306                MOV     R1,KIPDR3    ,LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
```

```
2890  026266  012737  026300  000004          MOV     #4$ ERRVEC      ,SET UP LOC  4 TO 4$ FOR ODD ADDR  ABORT
2891  026274  005237  060001                  INC     60001           ,CAUSE ODD ADDRESS ABORT THRU LOC  4
2892  026300  012706  001100          4$      MOV     #KERSTK,KSP     ,RESTORE THE STACK POINTER
2893  026304  013702  172306                  MOV     KIPDR3,R2       ,READ KIPDR3 INTO R2
2894  026310  020102                          CMP     R1,R2           ,WAS W-BIT LEFT CLEARED?
2895  026312  001401                          BEQ     5$              ,BRANCH IF YES
2896  026314  104025                          ERROR   25              ,W-BIT GOT SET DURING AN ODD ADDR  ABORT
2897                                                                  ,FOR TIGHTER SCOPE LOOP
2898                                                                  ,REPLACE ERROR CALL WITH
2899                                                                  ,"BR 3$" = 000757
2900  026316  012737  002332  000004  5$      MOV     #TIMERR,ERRVEC  ,RESTORE NORMAL CPU TRAP ROUTINE TO LOC 4
2901  026324  012737  026210  001110          MOV     #1$,$LPERR      ,RESET LOOP ON ERROR POINTER TO 1$
2902  026332  004737  036036                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2903
2904                                  ,,*****************************************************************
2905                                  ,*
2906                                  ,*      THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2907                                  ,*      TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2908                                  ,*      ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2909                                  ,*      VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR
2910                                  ,*      THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6.5>, AND <3 0>
2911                                  ,*      ARE CHECKED IN SR0   SO THE SR0 AND SR2 LOGIC AND THE
2912                                  ,*      KT ERROR LOGIC ARE CHECKED.
2913                                  ,*
2914                                  ,,*****************************************************************
2915
2916                                  ,,*****************************************************************
2917                                  ,*TEST 33       NON-RESIDENT ABORT TEST (ACF=0&4)
2918                                  ,*
2919                                  ,*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2920                                  ,*      LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2921                                  ,*      USER MODES.   PDR 4  IS LOADED WITH ACF'S = 0&4 AND
2922                                  ,*      THEN PHYSICAL ADDR  60000 IS ACCESSED TO CAUSE THE ABORT
2923                                  ,*
2924                                  ,,*****************************************************************
2925  026336  000004                  TST33   SCOPE
2926
2927  026340  012700  000600          1$      MOV     #600,R0         ,LOAD DATA FOR PAR'S INTO R0
2928  026344  010037  172346                  MOV     R0,KIPAR3       ,MAP KERNEL PAR'S 3&4 TO 12-16K
2929  026350  010037  172350                  MOV     R0,KIPAR4
2930  026354  010037  177646                  MOV     R0,UIPAR3       ,MAP USER PAR'S 3&4 TO 12-16K
2931  026360  010037  177650                  MOV     R0,UIPAR4
2932  026364  012737  077406  172306          MOV     #77406,KIPDR3   ;MAP KERNEL PDR 3  128 BLKS, READ-WRITE
2933  026372  012737  077406  177606          MOV     #77406,UIPDR3   ,MAP USER PDR 3  128 BLKS, READ-WRITE
2934  026400  012700  060000                  MOV     #60000,R0       ,LOAD VIRTUAL ADDR  TO REFERENCE PDR3 INTO R0
2935  026404  012701  100000                  MOV     #100000,R1      ,LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2936  026410  012703  100011                  MOV     #100011,R3      ,LOAD R3 WITH WHAT SR0 SHOULD READ - N R , KERNEL, PG 4
2937  026414  012702  077400                  MOV     #77400,R2       ,LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2938  026420  012737  026462  000250  2$      MOV     #5$,MMVEC       ,POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2939  026426  010237  172310                  MOV     R2,KIPDR4       ,LOAD ACF TEST VALUE INTO KIPDR4
2940  026432  010237  177610                  MOV     R2,UIPDR4       ,LOAD ACF TEST VALUE INTO UIPDR4
2941  026436  012737  026444  001110          MOV     #3$,$LPERR      ,SET LOOP ON ERROR POINTER TO 3$
2942  026444  006010                  3$      CLR     (R0)            ,CLEAR PHYS. LOC  60000 USING PDR3
2943  026446  013737  177776  001176          MOV     PSW,$TMP0       ,SAVE PSW IN CASE OF ERROR
2944  026454  005211                  4$      INC     (R1)            ,TRY TO REF  IT USING PDR4 - SHOULD TRAP TO 5$
2945  026456  104026                          ERROR   26              ,MEM  MGMT  ABORT DID NOT OCCUR
```

```
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ,REPLACE ERROR CALL WITH
                                                        ,"BR 3$" = 000772
026460  000425                        BR      8$      ,BRANCH AROUND STATUS REG  CHECKS IF NO ABORT
026462  062706  000004          5$    ADD     #4,SP   ,RESTORE STACK POINTER
026466  005710                        TST     (R0)    ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
026470  001401                        BEQ     6$      ,BRANCH IF YES
026472  104027                        ERROR   27      ,INSTRUCTION WAS NOT ABORTED, LOC  GOT CHANGED
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ,"BR 3$" = 000764
026474  013737  177572  001366  6$    MOV     SR0,WASSR0      ,READ STATUS REGISTER 0
026502  013737  177576  001370        MOV     SR2,WASSR2      ,READ STATUS REGISTER 2
026510  020337  001366                CMP     R3,WASSR0       ,DID SR0 REPORT NON-RESIDENT ERROR CORRECTLY?
026514  001401                        BEQ     7$              ;BRANCH IF YES
026516  104030                        ERROR   30              ,SR0 DID NOT REPORT NON-RES  ERROR CORRECTLY
                                                        ,FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ,"BR 3$" = 000752
026520  012704  026454          7$    MOV     #4$,R4          ,LOAD R4 WITH WHAT SR2 SHOULD READ
026524  020437  001370                CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR  (=4$)?
026530  001401                        BEQ     8$              ;BRANCH IF YES
026532  104031                        ERROR   31              ;SR2 DID NOT LOCK VIRTUAL ADDR  OF NON-RES  ERROR
                                                        ,FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ,"BR 3$" = 000744
026534  042737  160000  177572  8$    BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
026542  032737  140000  001176        BIT     #140000,$TMP0   ;HAS ACF=0&4 BEEN TESTED IN USER YET
026550  001006                        BNE     9$              ,BRANCH IF YES
026552  012703  100151                MOV     #100151,R3      ,LOAD R3 WITH WHAT SR0 SHOULD READ - N R , USER, PG 4
026556  012737  140000  177776        MOV     #140000,PSW     ,GO TO USER MODE
026564  000715                        BR      2$              ;REPEAT TEST IN USER MODE
026566  022702  077404          9$    CMP     #77404,R2       ;HAS ACF=4 BEEN TESTED YET?
026572  001407                        BEQ     10$             ;BRANCH IF YES
026574  012702  077404                MOV     #77404,R2       ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
026600  012703  100011                MOV     #100011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ-N R ,KERNEL,PG  4
026604  005037  177776                CLR     PSW             ;GO BACK TO KERNEL MODE
026610  000703                        BR      2$              ,GO BACK & TEST ACF=4 IN SAME MODE
026612  005037  177776          10$   CLR     PSW             ,GO BACK TO KERNEL MODE BEFORE LEAVING
026616  012737  026340  001110        MOV     #1$,$LPERR      ,RESET LOOP ON ERROR POINTER TO 1$
026624  012737  002400  000250        MOV     #MGMERR,MMVEC   ,RESTORE ADDRESS OF NORMAL MEMORY
                                                        ;MANAGEMENT ERROR ROUTINE TO MMVEC

                                ,,**********************************************************************
                                ,*TEST 34       READ-ONLY ABORT TEST (ACF=2)
                                ,*
                                ,*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
                                ,*      LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
                                ,*      USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
                                ,*      PHYSICAL ADDR  60000 IS WRITTEN TO CAUSE THE ABORT
                                ,*
                                ,,**********************************************************************
026632  000004                  TST34   SCOPE
026634                          1$                              ,KERNEL & USER PAR'S 3 & 4 AND PDR 3
                                                                ,ARE SETUP FROM LAST TEST
026634  012700  060000                  MOV     #60000,R0       ,LOAD VIRTUAL ADDR  TO REFERENCE PDR3 INTO R0
```

```
3002  026640  012701  100000                    MOV     #100000,R1      ;LOAD VIRTUAL ADDR  TO REFERENCE PDR4 INTO R1
3003  026644  012703  020011                    MOV     #20011,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG 4
3004  026650  012702  077402                    MOV     #77402,R2       ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
3005  026654  012737  026716  000250  2$        MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
3006  026662  010237  172310                    MOV     R2,KIPDR4       ;LOAD ACF=2 INTO KIPDR4
3007  026666  010237  177610                    MOV     R2,UIPDR4       ;LOAD ACF=2 INTO UIPDR4
3008  026672  012737  026700  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3009  026700  005010                    3$      CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
3010  026702  013737  177776  001176            MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
3011  026710  005211                    4$      INC     (R1)            ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
3012  026712  104026                            ERROR   26              ;MEM. MGMT. ABORT DID NOT OCCUR
3013                                                                    ;FOR TIGHTER SCOPE LOOP
3014                                                                    ;REPLACE ERROR CALL WITH
3015                                                                    ;"BR 3$" = 000772
3016  026714  000425                            BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
3017  026716  062706  000004            5$      ADD     #4,SP           ;RESTORE STACK POINTER
3018  026722  005710                            TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
3019  026724  001401                            BEQ     6$              ;BRANCH IF YES
3020  026726  104027                            ERROR   27              ;INSTRUCTION WAS NOT ABORTED, LOC  GOT CHANGED
3021                                                                    ;FOR TIGHTER SCOPE LOOP
3022                                                                    ;REPLACE ERROR CALL WITH
3023                                                                    ;"BR 3$" = 000764
3024  026730  013737  177572  001366  6$        MOV     SR0,WASSR0      ;READ STATUS REG. 0
3025  026736  013737  177576  001370            MOV     SR2,WASSR2      ;READ STATUS REG. 2
3026  026744  020337  001366                    CMP     R3,WASSR0       ;DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
3027  026750  001401                            BEQ     7$              ;BRANCH IF YES
3028  026752  104030                            ERROR   30              ;SR0 DID NOT REPORT R/O ERROR CORRECTLY
3029                                                                    ;FOR TIGHTER SCOPE LOOP
3030                                                                    ;REPLACE ERROR CALL WITH
3031                                                                    ;"BR 3$" = 000752
3032  026754  012704  026710            7$      MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
3033  026760  020437  001370                    CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR  (=4$)?
3034  026764  001401                            BEQ     8$              ;BRANCH IF YES
3035  026766  104031                            ERROR   31              ;SR2 DID NOT LOCKUP VIRTUAL ADDR  OF R/O ERROR
3036                                                                    ;FOR TIGHTER SCOPE LOOP
3037                                                                    ;REPLACE ERROR CALL WITH
3038                                                                    ;"BR 3$" = 000744
3039  026770  042737  160000  177572  8$        BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
3040  026776  032737  140000  001176            BIT     #140000,$TMP0   ;HAS ACF=2 BEEN TESTED IN USER MODE?
3041  027004  001006                            BNE     9$              ;BRANCH IF YES
3042  027006  012703  020151                    MOV     #20151,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG 4
3043  027012  012737  140000  177776            MOV     #140000,PSW     ;GO TO USER MODE
3044  027020  000715                            BR      2$              ;REPEAT TEST IN USER MODE
3045  027022  005037  177776            9$      CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
3046  027026  012737  026634  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3047  027034  012737  002400  000250            MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
3048                                                                    ;MANAGEMENT ERROR ROUTINE TO MMVEC
3049                                    ;;**************************************************************
3050                                    ;*TEST 35        TEST ILLEGAL MODE "01"
3051                                    ;*
3052                                    ;*      THIS TEST CHECKS TO SEE THAT A 01 IN THE CURRENT MODE BITS OF THE
3053                                    ;*      PSW WHILE MEMORY MANAGEMENT IS ON IS ILLEGAL.  A
3054                                    ;*      MEMORY MANAGEMENT ABORT SHOULD OCCUR AND STATUS REGISTER 0
3055                                    ;*      SHOULD REPORT NON-RESIDENT ABORT, MODE = 01, PAGE = 1 (100043)   STATUS
3056                                    ;*      REGISTER 2 SHOULD LOCKUP THE ADDRESS OF THE INSTRUCTION
3057                                    ;*      THAT LOADED THE PSW
```

```
3058                                    ;*
3059                                    ;;*****************************************************************
3060  027042  000004                    TST35   SCOPE
3061  027044  012737  027060  001110    1$      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3062  027052  012737  027074  000250            MOV     #3$,MMVEC       ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
3063  027060  012737  040000  177776    2$      MOV     #40000,PSW      ;SET 01 IN PSW CURRENT MODE BITS
3064  027066  000240                            NOP                     ;FETCH OF THIS INSTRUCTION SHOULD CAUSE ABORT
3065  027070  104056                            ERROR   56              ;ILLEGAL MODE 01 NOT ABORTED
3066                                                                    ;FOR A TIGHTER SCOPE LOOP
3067                                                                    ;REPLACE ERROR CALL WITH
3068                                                                    ;"BR 2$" = 000773
3069  027072  000424                            BR      5$              ;BRANCH AROUND SR0 & SR2 CHECKS
3070  027074  012706  001100            3$      MOV     #KERSTK,SP      ;RESTORE STACK POINTER
3071  027100  012701  100043                    MOV     #100043,R1      ;LOAD EXPECTED CONTENTS OR SR0 INTO R1
3072  027104  013737  177572  001366            MOV     SR0,WASSR0      ;READ CONTENTS OF SR0
3073
3074  027112  020137  001366                    CMP     R1,WASSR0       ;DID SR0 REPORT ILLEGAL MODE CORRECTLY?
3075  027116  001401                            BEQ     4$              ;BRANCH IF YES
3076  027120  104057                            ERROR   57              ;SR0 DID NOT REPORT NR ABORT, PG=1, MODE=01
3077                                                                    ;FOR TIGHTER SCOPE LOOP
3078                                                                    ;REPLACE ERROR CALL WITH
3079                                                                    ;"BR 2$" = 000757
3080  027122  012701  027060            4$      MOV     #2$,R1          ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
3081  027126  013737  177576  001370            MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
3082  027134  020137  001370                    CMP     R1,WASSR2       ;DID SR2 LOCKUP VIRT  ADDR OF ABORTED INST
3083  027140  001401                            BEQ     5$              ;BRANCH IF YES
3084  027142  104036                            ERROR   36              ;SR2 DID NOT LOCKUP VIRT  ADDR  OF ILL  MODE INST
3085                                                                    ;FOR TIGHTER SCOPE LOOP
3086                                                                    ;REPLACE ERROR CALL WITH
3087                                                                    ;"BR 2$" = 000746
3088  027144  042737  160000  177572    5$      BIC     #160000,SR0     ;CLEAR POSSIBLE ERROR BITS IN SR0
3089  027152  012737  002400  000250            MOV     #MGMERR,MMVEC   ;RESTORE MEM  MGMT  ABORT VECTOR
3090  027160  012737  027044  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103                                    ;;*****************************************************************
3104                                    ;*
3105                                    ;*      THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
3106                                    ;*      COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
3107                                    ;*      AND STATUS LOGIC.  THE PAGE LENGTH FIELD (PLF) IN KERNEL
3108                                    ;*      PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
3109                                    ;*      ADDRESSES ARE READ.  WHILE USING BOTH UPWARD & DOWNWARD PAGE
3110                                    ;*      EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
3111                                    ;*      "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T
3112                                    ;*
3113                                    ;*      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).
                        ;*
                        ;;**********************************************************************
027166  000004          TST36.  SCOPE
027170  012737  077406  172306  1$:     MOV     #77406,KIPDR3   ;MAKE SURE PDR3 IS DESCRIBED AS R/W
027176  012737  077406  172312          MOV     #77406,KIPDR5   ;MAKE SURE PDR5 IS DESCRIBED AS R/W
027204  012700  100000          2$:     MOV     #100000,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
027210  012704  000406                  MOV     #406,R4         ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
027214  012737  027406  000250          MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
027222  010437  172310                  MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
027226  012737  027234  001110          MOV     #3$,SLPERR      ;SET LOOP ON ERROR POINTER TO 3$
027234  012706  001100          3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
027240  011001                          MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
027242  062700  000100                  ADD     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
027246  012737  027254  001110          MOV     #4$,SLPERR      ;SET LOOP ON ERROR POINTER TO 4$
027254  012706  001100          4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
027260  011001                          MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
027262  062700  000100                  ADD     #100,R0         ;FORM NEXT VIRTUAL ADDR IN R0
027266  020027  117700                  CMP     R0,#117700      ;HAVE ALL PLF'S BEEN TESTED YET?
027272  001470                          BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
027274  012737  027310  001110          MOV     #5$,SLPERR      ;SET LOOP ON ERROR POINTER TO 5$
027302  012737  027316  000250          MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
027310  011001                  5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
027312  104033                          ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000776
027314  000424                          BR      8$              ;BRANCH AROUND ABORT CHECKS
027316  012706  001100          6$      MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
027322  013737  177572  001366          MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
027330  013737  177576  001370          MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
027336  012702  040011                  MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
027342  020237  001366                  CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
027346  001401                          BEQ     7$              ;BRANCH IF YES
027350  104034                          ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000757
027352  012703  027310          7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
```

```
3170  027356  020337  001370            CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT  ADDR  OF ABORTED INSTRUCTION?
3171  027362  001401                    BEQ     8$              ;BRANCH IF YES
3172  027364  104035                    ERROR   35              ;SR2 DID NOT LOCKUP VIRT  ADDR  OF ABORT CORRECTLY
3173                                                            ;FOR TIGHTER SCOPE LOOP
3174                                                            ;REPLACE ERROR CALL WITH
3175                                                            ;"BR 5$" = 000751
3176  027366  042737  160000  177572  8$      BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3177  027374  062704  000400                  ADD     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
3178  027400  162700  000100                  SUB     #100,R0         ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
3179  027404  000703                    BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3180                                                            ;THE PLF VALUE JUST FORMED
3181  027406  012637  001356          9$.     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3182  027412  012637  001360                  MOV     (KSP)+,TRAPPS
3183  027416  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
3184  027424  013737  177576  001370          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3185  027432  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3186  027440  104032                    ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3187                                                            ;FOR TIGHTER SCOPE LOOP
3188                                                            ;REPLACE ERROR CALL WITH
3189                                                            ;A "NOP" = 000240
3190  027442  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3191  027446  013746  001356                  MOV     TRAPPC,-(KSP)
3192  027452  000002                    RTI                     ;RETURN FROM UNEXPECTED ABORT
3193
3194  027454  012737  027170  001110  10$.     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3195  027462  012737  002400  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M.  TRAP HANDLER
3196                                                            ;ADDRESS TO M.M.  TRAP VECTOR
3197
3198
3199                                    ;;**********************************************************************
3200                                    ;*TEST 37       PAGE LENGTH FAULTS-DOWNWARD EXPANSION
3201                                    ;*
3202                                    ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
3203                                    ;*      FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
3204                                    ;*      ARE ACCESSED.  WHEN VBA (12:6) IS GREATER THAN OR EQUAL TO PDR (14 8)
3205                                    ;*      NO PAGE ABORT SHOULD OCCUR.  WHEN VBA (12:6) IS LESS THAN PDR (14 8)
3206                                    ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
3207                                    ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
3208                                    ;*      (BIT 3) OF PDR4=1).
3209                                    ;*
3210                                    ;;**********************************************************************
3211  027470  000004                    TST37:  SCOPE
3212  027472  012700  117700            1$:     MOV     #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3213  027476  012704  077016                    MOV     #77016,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
3214  027502  012737  027674  000250  2$:      MOV     #9$,MMVEC       ;SETUP M.M.  TRAP VECTOR FOR UNEXPECTED ABORTS
3215  027510  010437  172310                  MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3216  027514  012737  027522  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3217  027522  012706  001100          3$.     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3218  027526  011001                    MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
3219  027530  162700  000100                  SUB     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
3220  027534  012737  027542  001110          MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3221  027542  012706  001100          4$      MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3222  027546  011001                    MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3223  027550  162700  000100                  SUB     #100,R0         ;FORM NEXT VIRTUAL ADDR.  IN R0
3224  027554  020027  100000                  CMP     R0,#100000      ;HAVE ALL PLF'S BEEN TESTED YET?
3225  027560  001470                    BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
```

```
3226  027562  012737  027576  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3227  027570  012737  027604  000250          MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3228  027576  011001                  5$      MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
3229  027600  104033                          ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3230                                                                  ;FOR TIGHTER SCOPE LOOP
3231                                                                  ;REPLACE ERROR CALL WITH
3232                                                                  ;"BR 5$" = 000776
3233  027602  000424                          BR      8$              ;BRANCH AROUND ABORT CHECKS
3234  027604  012706  001100          6$      MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
3235  027610  013737  177572  001366          MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
3236  027616  013737  177576  001370          MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
3237  027624  012702  040011                  MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
3238  027630  020237  001366                  CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
3239  027634  001401                          BEQ     7$              ;BRANCH IF YES
3240  027636  104034                          ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3241                                                                  ;FOR TIGHTER SCOPE LOOP
3242                                                                  ;REPLACE ERROR CALL WITH
3243                                                                  ;"BR 5$" = 000757
3244  027640  012703  027576          7$      MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
3245  027644  020337  001370                  CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT ADDR OF ABORTED INSTRUCTION?
3246  027650  001401                          BEQ     8$              ;BRANCH IF YES
3247  027652  104035                          ERROR   35              ;SR2 DID NOT LOCKUP VIRT ADDR or ABORT CORRECTLY
3248                                                                  ;FOR TIGHTER SCOPE LOOP
3249                                                                  ;REPLACE ERROR CALL WITH
3250                                                                  ;"BR 5$" = 000751
3251  027654  042737  160000  177572  8$      BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3252  027662  162704  000400                  SUB     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
3253  027666  062700  000100                  ADD     #100,R0         ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
3254  027672  000703                          BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3255                                                                  ;THE PLF VALUE JUST FORMED
3256  027674  012637  001356          9$      MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3257  027700  012637  001360                  MOV     (KSP)+,TRAPPS
3258  027704  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
3259  027712  013737  177576  001370          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3260  027720  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3261  027726  104032                          ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3262                                                                  ;FOR TIGHTER SCOPE LOOP
3263                                                                  ;REPLACE ERROR CALL WITH
3264                                                                  ;A "NOP" = 000240
3265  027730  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3266  027734  013746  001356                  MOV     TRAPPC,-(KSP)
3267  027740  000002                          RTI                     ;RETURN FROM UNEXPECTED ABORT
3268
3269  027742  012737  027472  001110  10$     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3270  027750  012737  002400  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3271                                                                  ;ADDRESS TO M.M. TRAP VECTOR
3272
3273
3274                                  ;**********************;**********************************************
3275                                  ;*TEST 40        SR2 BIT TST
3276                                  ;*
3277                                  ;*      THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
3278                                  ;*      CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 100000
3279                                  ;*      TO 111000 (PHYSICAL ADDRESSES 060000-071000)   KIPDR4 IS USED TO EXECUTE
3280                                  ;*      THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY
3281                                  ;*      010727  MOV     PC,(PC)+  ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
```

```
3282                                    ;*      000000                  ;ITS VIRTUAL ADDR  SHOULD BE LOCKED UP IN SR2
3283                                    ;*      000137  JMP     @#3$    ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
3284                                    ;*      (ADDR  OF 3$)           ;IN CASE A R/O ABORT DOES NOT OCCUR,
3285                                    ;*                              ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR
3286                                    ;*
3287                                    ;;**********************************************************************
3288  027756  000004                    TST40   SCOPE
3289  027760  012737  000600  172346    1$      MOV     #600,KIPAR3     ;BE SURE PAR3 IS MAPPED TO 12-16K
3290  027766  012737  000600  172350            MOV     #600,KIPAR4     ;BE SURE PAR4 IS MAPPED TO 12-16K
3291  027774  012737  077406  172306            MOV     #77406,KIPDR3   ;MAP PAGE 3  128 BLOCKS, R/W
3292  030002  012737  077402  172310            MOV     #77402,KIPDR4   ;MAP PAGE 4  128 BLOCKS, READ-ONLY
3293  030010  012700  060000                    MOV     #60000,R0       ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
3294  030014  012701  100000                    MOV     #100000,R1      ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
3295  030020  012737  030054  000250            MOV     #3$,MMVEC       ;SET M.M.  TRAP VECTOR TO 3$
3296  030026  012737  030034  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3297  030034  012720  010727            2$      MOV     #010727,(R0)+   ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR
3298  030040  005020                            CLR     (R0)+           ;       REACHED THRU PDR/PAR 4.
3299  030042  012720  000137                    MOV     #000137,(R0)+   ;LOAD "JMP @#3$" INSTRUCTION AT VIRT. ADDR
3300  030046  012710  030054                    MOV     #3$,(R0)        ;       IN CASE R/O VIOL. DOES NOT ABORT
3301  030052  010107                            MOV     R1,PC           ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3302  030054  012706  001100            3$      MOV     #KERSTK,KSP     ;RESTORE STACK POINTER
3303  030060  013737  177576  001370            MOV     SR2,WASSR2      ;READ CONTENTS OF STATUS REG 2
3304  030066  020137  001370                    CMP     R1,WASSR2       ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
3305  030072  001401                            BEQ     4$              ;BRANCH IF YES
3306  030074  104036                            ERROR   36              ;SR2 DID NOT LOCK UP VIRTUAL ADDR  OF R/O VIOL
3307                                                                    ;FOR TIGHTER SCOPE LOOP
3308                                                                    ;REPLACE ERROR CALL WITH
3309                                                                    ;"BR 2$" = 000757
3310  030076  042737  160000  177572    4$      BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
3311  030104  162700  000004                    SUB     #4,R0           ;RESET R0 TO POINT TO NEXT VIRT  ADDR  TO LOAD
3312  030110  062701  000002                    ADD     #2,R1           ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
3313  030114  020127  111002                    CMP     R1,#111002      ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
3314  030120  103745                            BLO     2$              ;BRANCH IF NO
3315
3316  030122  012737  027760  001110    5$      MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3317  030130  012737  077406  172310            MOV     #77406,KIPDR4   ;RESTORE PDR4 TO R/W ACCESS
3318  030136  012737  002400  000250            MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL M.M
3319                                                                    ;TRAP HANDLER TO M.M.  VECTOR
3320
3321
```

```
                                        ;*****************************************************************
                                        ;*TEST 41       MORE CHECKS OF SR0 & SR2
                                        ;*
                                        ;*      THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SR0 & SR2 LOGIC
                                        ;*      FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS
                                        ;*      PROGRAM COUNTER.  ALSO SR0 & SR2 ARE LOCKED UP BY A PAGE LENGTH
                                        ;*      ABORT.  THEN WITHOUT CLEARING SR0'S ERROR BITS, A R/O ABORT IS CAUSED
                                        ;*      SR0 & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
                                        ;*      INFORMATION ABOUT THE PAGE LENGTH ABORT SOULD STILL BE LOCKED UP
                                        ;*      IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SR0 IS CLEARED
                                        ;*      AND SR2 IS UNLOCKED BY A RESET.  AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
                                        ;*      SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
                                        ;*
                                        ;*****************************************************************
030144  000004                TST41   SCOPE
030146  012737  000600  172352  1$    MOV     #600,KIPAR5     ;MAP KERNEL PAGE 5 TO 12-16K
030154  012737  000406  172310        MOV     #406,KIPDR4     ;SETUP PDR4 FOR PAGE LENGTH ABORT
030162  012737  077402  172312        MOV     #77402,KIPDR5   ;SETUP PDR5 FOR R/O ABORT
030170  012737  030176  001110        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
030176  013737  177576  001370  2$    MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
030204  012701  030176                MOV     #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
030210  020137  001370                CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
030214  001401                        BEQ     3$              ;BRANCH IF YES
030216  104041                        ERROR   41              ;SR2 NOT TRACKING CORRECTLY
                                                            ;FOR TIGHTER SCOPE LOOP
                                                            ;REPLACE ERROR CALL WITH
                                                            ;"BR 2$" = 000767
030220  012737  030226  001110  3$    MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
030226  013737  177576  001370  4$    MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
030234  012701  030226                MOV     #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
030240  020137  001370                CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 4$
030244  001401                        BEQ     5$              ;BRANCH IF YES
030246  104041                        ERROR   41              ;SR2 NOT TRACKING CORRECTLY
                                                            ;FOR TIGHTER SCOPE LOOP
                                                            ;REPLACE ERROR CALL WITH
                                                            ;"BR 4$" = 000767
030250  012737  030256  001110  5$    MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
030256  012737  030274  000250  6$    MOV     #7$,MMVEC       ;PUT ADDRESS OF 7$ IN M.M  TRAP VECTOR
030264  005037  001200                CLR     $TMP1           ;CLEAR ERROR INDICATOR
030270  005237  100500                INC     @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
030274  012706  001100          7$    MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
030300  013737  177572  001176        MOV     SR0,$TMP0       ;SAVE SR0'S INFORMATION ON PG. LGTH  ABORT
030306  013737  177576  001202        MOV     SR2,$TMP2       ;SAVE SR2'S INFORMATION ON PG. LGTH  ABORT
030314  012737  030326  000250        MOV     #8$,MMVEC       ;PUT ADDRESS OF 8$ IN M.M.  TRAP VECTOR
030322  005237  120000                INC     @#120000        ;CAUSE R/O ABORT - TRAP TO 8$
030326  012706  001100          8$    MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
030332  013737  177572  001366        MOV     SR0,WASSR0      ;READ SR0 FOLLWOING SECOND KT ABORT
030340  013737  177576  001370        MOV     SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
030346  023737  001176  001366        CMP     $TMP0,WASSR0    ;IS SR0 STILL HOLDING INFO ON FIRST ABORT?
030354  001402                        BEQ     9$              ;BRANCH IF YES
030356  005237  001200                INC     $TMP1           ;SET ERROR INDICATOR
030362  023737  001202  001370  9$    CMP     $TMP2,WASSR2    ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
030370  001402                        BEQ     10$             ;BRANCH IF YES
030372  005237  001200                INC     $TMP1           ;SET ERROR INDICATOR
```

```
3378  030376  005737  001200          10$     TST     $TMP1           ;WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
3379  030402  001401                          BEQ     11$             ;BRANCH IF NO
3380  030404  104037                          ERROR   37              ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3381                                                                  ;FOR TIGHTER SCOPE LOOP
3382                                                                  ;REPLACE ERROR CALL WITH
3383                                                                  ;"BR 6$" = 000726
3384  030406  005037  001200          11$     CLR     $TMP1           ;CLEAR ERROR INDICATOR
3385  030412  000005                          RESET                   ;EXECUTE A RESET, APPLYING AN "INIT"
3386  030414  013737  177572  001366          MOV     SR0,WASSR0      ;READ SR0
3387  030422  005737  001366                  TST     WASSR0          ;WAS SR0 CLEARED BY THE RESET?
3388  030426  001402                          BEQ     12$             ;BRANCH IF YES
3389  030430  005237  001200                  INC     $TMP1           ;SR0 NOT CLEARED BY A RESET
3390  030434  013737  177576  001370  12$     MOV     SR2,WASSR2      ;READ SR2
3391  030442  022737  030434  001370          CMP     #12$,WASSR2     ;WAS SR2 UNLOCKED BY A RESET?
3392  030450  001402                          BEQ     13$             ;BRANCH IF YES
3393  030452  005237  001200                  INC     $TMP1           ;SR2 NOT UNLOCKED BY A RESET
3394  030456  005737  001200          13$     TST     $TMP1           ;WERE SR0 & SR2 BOTH "RESET" BY A RESET?
3395  030462  001401                          BEQ     14$             ;BRANCH IF YES
3396  030464  104040                          ERROR   40              ;SR0 OR SR2 NOT "RESET" BY A RESET
3397                                                                  ;FOR TIGHTER SCOPE LOOP
3398                                                                  ;REPLACE ERROR CALL WITH
3399                                                                  ;"BR 6$" = 000676
3400  030466  005237  177572          14$     INC     SR0             ;TURN MEMORY MANAGEMENT BACK ON
3401  030472  013737  177576  001370  15$     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3402  030500  012701  030472                  MOV     #15$,R1         ;PUT EXPECTED VIRTUAL PC IN R1
3403  030504  020137  001370                  CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3404  030510  001401                          BEQ     16$             ;BRANCH IF YES
3405  030512  104041                          ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3406                                                                  ;FOR TIGHTER SCOPE LOOP
3407                                                                  ;REPLACE ERROR CALL WITH
3408                                                                  ;"BR 6$" = 000663
3409                                  ;*****************************************************************
3410
3411                                  ;       ALL CODE WITH A ';+ ' DESIGNATION IN THE COMMENT INDICATES THAT IT HAS BEEN
3412                                  ;       ADDED TO THE ORIGINAL REV OF THIS DIAGNOSTIC THESE ENHANCEMENTS OR CORRECTIONS
3413                                  ;        ARE MADE BY THE PRODUCT ENHANCEMENT GROUP (DIAGNOSTICS)
3414
3415                                  ;*****************************************************************
3416
3417
3418
3419  030514  012737  030574  000250  16$     MOV     #21$,MMVEC      ;+SET UP KTVEC FOR NON-RESIDENT ABORT
3420  030522  012737  077400  172312          MOV     #77400,KIPDR5   ;+SET UP KIPDR5 FOR NR ABORT
3421  030530  012737  030536  001110          MOV     #17$,$LPERR     ;+SET LOOP ON ERROR POINTER
3422  030536  005037  001200          17$     CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3423  030542  005237  130000          20$     INC     @#130000        ;+CAUSE A NON-RESIDENT ABORT
3424  030546  005237  001200                  INC     $TMP1           ;+SET ERROR INDICATOR
3425  030552  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3426  030560  013737  177576  001370          MOV     SR2,WASSR2      ;+READ SR2
3427  030566  104065                          ERROR   65              ;+NON-RESIDENT ABORT DID NOT OCCUR
3428
3429
3430  030570  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3431  030574  012706  001100          21$     MOV     #KERSTK,KSP     ;+RESTORE KERNEL STACK POINTER
3432  030600  005737  177572          22$     TST     SR0             ;+TEST FOR THE NR ABORT IN SR0 (BIT 15)
3433  030604  100413                          BMI     23$             ;+IF SET BRANCH
```

```
3434  030606  005237  001200                  INC     $TMP1           ;+SET ERROR INDICATOR
3435  030612  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3436  030620  013737  177576  001370          MOV     SR2,WASSR2      ;+READ SR2
3437  030626  104066                          ERROR   66              ;+NR ABORT ERROR DID NOT SET IN SR0 (BIT 15)
3438
3439
3440  030630  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3441  030634  023727  177576  030542  23$     CMP     SR2,#20$        ;+TEST THAT SR2 FROZE TO THE VIRTUAL
3442                                                                  ;+ADDRESS OF THE NR ABORT INSTR
3443  030642  001413                          BEQ     24$             ;+IF EQUAL BRANCH
3444  030644  005237  001200                  INC     $TMP1           ;+ELSE ERROR
3445  030650  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3446  030656  013737  177576  001370          MOV     SR2,WASSR2      ;+READ SR2
3447  030664  104067                          ERROR   67              ;+SR2 DID NOT CONTAIN THE CORRECT ADDRESS
3448
3449
3450  030666  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3451  030672  012737  030732  000250  24$     MOV     #25$,MMVEC      ;+SET KTVEC FOR 2ND NR ABORT
3452  030700  005237  130000                  INC     @#130000        ;+CAUSE ABORT
3453  030704  005237  001200                  INC     $TMP1           ;+SET ERROR INDICATOR
3454  030710  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3455  030716  013737  177576  001370          MOV     SR2,WASSR2      ;+READ SR2
3456  030724  104070                          ERROR   70              ;+2ND NR ABORT DID NOT OCCUR
3457
3458
3459  030726  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3460  030732  012706  001100          25$     MOV     #KERSTK,KSP     ;+RESTORE KERNEL STACK
3461  030736  013737  177576  001370          MOV     SR2,WASSR2      ;+READ SR2
3462  030744  023727  001370  030542          CMP     WASSR2,#20$     ;+TEST SR2 FOR VIRTUAL ADDRESS OF
3463                                                                  ;+OF THE FIRST NON-RESIDENT ABORT
3464  030752  001413                          BEQ     26$             ;+IF EQUAL BRANCH
3465                                                                  ;+ELSE ERROR
3466  030754  005237  001200                  INC     $TMP1           ;+SET ERROR INDICATOR
3467  030760  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3468  030766  012737  030542  001364          MOV     #20$,CORSR2     ;+READ CORRECT SR2 VALUE
3469  030774  104071                          ERROR   71              ;+SR2 DID NOT CONTAIN THE VALUE OF
3470                                                                  ;+THE FIRST NR ABORT
3471
3472
3473  030776  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3474  031002  000005                  26$     RESET                   ;+ISSUE INIT  CONDITIONS
3475  031004  005737  177572                  TST     SR0             ;+SR0 SHOULD BE CLEAR
3476  031010  001412                          BEQ     27$             ;+IF TRUE BRANCH
3477                                                                  ;+ELSE ERROR
3478  031012  005237  001200                  INC     $TMP1           ;+SET ERROR INDICATOR
3479  031016  013737  177572  001366          MOV     SR0,WASSR0      ;+READ SR0
3480  031024  005037  001362                  CLR     CORSR0          ;+SET CORRECT SR0 VALUE FOR ERROR MSG
3481  031030  104072                          ERROR   72              ;+SR0 DID NOT CLEAR AFTER INIT
3482
3483
3484
3485  031032  005037  001200                  CLR     $TMP1           ;+CLEAR ERROR INDICATOR
3486  031036  005237  177572          27$     INC     SR0             ;+TURN MEM MANG ON
3487  031042  013737  177576  001370  30$     MOV     SR2,WASSR2      ;+READ SR2
3488  031050  023727  001370  031042          CMP     WASSR2,#30$     ;+IS SR2 TRACKING AGAIN
3489  031056  001405                          BEQ     31$             ;+YES, BRANCH
```

```
3490
3491  031060  005237  001200                  INC     $TMP1           ;+ELSE ERROR
3492  031064  012701  031042                  MOV     #30$,R1         ;+SET ERROR INDICATOR
3493  031070  104041                          ERROR   41              ;+SET CORRECT SR2 VALUE FOE ERROR MSG
3494                                                                  ;+ERROR, SR2 IS NOT TRACKING CORRECTLY
3495
3496  031072  012737  030146  001110  31$     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3497  031100  012737  077406  172310          MOV     #77406,KIPDR4   ;RESET PDR4 TO 128 BLKS, R/W
3498  031106  012737  077406  172312          MOV     #77406,KIPDR5   ;RESET PDR5 TO 128 BLKS, R/W
3499  031114  012737  002400  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
3500                                                                  ;MANAGEMENT TRAP ROUTINE TO M M  VECTOR
3501
3502
3503                                  ;;*****************************************************************
3504                                  ;*TEST 42       USER ABORT PICKS UP KERNEL SPACE VECTOR
3505                                  ;*
3506                                  ;*      THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
3507                                  ;*      USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
3508                                  ;*      KERNEL SPACE.  USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
3509                                  ;*      THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
3510                                  ;*      WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
3511                                  ;*      SHOULD BE PICKED UP FROM LOC  000004.  AN ODD ADDRESS ERROR IS USED
3512                                  ;*      TO CAUSE A TRAP TO "4"
3513                                  ;*
3514                                  ;;*****************************************************************
3515  031122  000004                  TST42   SCOPE
3516  031124  004737  036002          1$      JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
3517  031130  012737  031136  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3518  031136  005037  177776          2$      CLR     PSW             ;GO TO KERNEL MODE
3519  031142  012706  001100                  MOV     #KERSTK,KSP     ;SETUP KERNEL STACK PTR
3520  031146  012737  000600  177640          MOV     #600,UIPAR0     ;MAP USER PAGE 0 TO 12K
3521  031154  012737  031236  000004          MOV     #4$,@#4         ;LOAD KERNEL VECTOR 4 (LOC 4) WITH 4$
3522  031162  012737  000340  000006          MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
3523  031170  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
3524  031176  012706  000700                  MOV     #USESTK,USP     ;SETUP USER STACK PTR
3525  031202  012737  031222  000004          MOV     #3$,@#4         ;LOAD USER VECTOR 4 (LOC  60004) WITH 3$
3526  031210  012737  000340  000006          MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
3527  031216  005737  031223                  TST     3$+1            ;CAUSE ODD ADDR. ERROR TRAP TO "4"
3528                                                                  ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
3529                                                                  ;LOC.  4, NOT PC=3$ FROM USER LOC  4 (=60004)
3530  031222  013701  177776          3$      MOV     PSW,R1          ;SAVE PSW FOR ERROR
3531  031226  010602                          MOV     SP,R2           ;SAVE VALUE OF STACK POINTER FOR ERROR
3532  031230  005037  177776                  CLR     PSW             ;BE SURE BACK IN KERNEL MODE
3533  031234  104042                          ERROR   42              ;DID NOT TRAP THRU KERNEL SPACE
3534                                                                  ;FOR TIGHTER SCOPE LOOP
```

```
3535                                                            ,REPLACE ERROR CALL WITH
3536                                                            ,"BR 2$" = 000740
3537  031236  005037  177776          4$      CLR     PSW       ,BE SURE BACK IN KERNEL MODE
3538  031242  012706  001100                  MOV     #KERSTK,KSP ,RESTORE KERNEL S.P  IN CASE IT CHANGED
3539  031246  005037  177640                  CLR     UIPAR0    ;REMAP USER PAGE 0 TO 0-4K
3540  031252  012737  140000  177776          MOV     #140000,PSW ,GO TO USER MODE
3541  031260  012706  000700                  MOV     #USESTK,USP ,RESTORE USER STACK POINTER
3542  031264  005037  177776                  CLR     PSW       ,GO BACK TO KERNEL MODE
3543  031270  012737  002332  000004          MOV     #TIMERR,@#4 ,RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3544  031276  012737  031124  001110          MOV     #1$,$LPERR ,RESET LOOP ON ERROR POINTER TO 1$
3545  031304  004737  036036                  JSR     PC,TON    ,TURN T-BIT TRAPPING BACK ON
3546
3547                                  ,,**********************************************************************
3548                                  ,*TEST 43       RTI IN USER MODE DOES NOT CHANGE PSW
3549                                  ,*
3550                                  ,*      THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3551                                  ,*      MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED
3552                                  ,*
3553                                  ,,**********************************************************************
3554  031310  000004                  TST43   SCOPE
3555
3556  031312  012737  031324  001110  1$      MOV     #2$,$LPERR ,SET LOOP ON ERROR POINTER TO 2$
3557  031320  012702  170000                  MOV     #170000,R2 ,LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
3558  031324  010237  177776          2$      MOV     R2,PSW    ,GO TO USER MODE-PRIORITY 0
3559  031330  012746  000340                  MOV     #340,-(SP) ,PUT A NEW PSW (PRIORITY=7) ON STACK
3560  031334  012746  031342                  MOV     #3$,-(SP) ,PUT NEW PC ON THE STACK
3561  031340  000002                          RTI               ,DO AN RTI FROM USER MODE
3562  031342  013701  177776          3$      MOV     PSW,R1    ,READ NEW PSW INTO R1
3563  031346  042701  007437                  BIC     #7437,R1  ,MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3564  031352  005037  177776                  CLR     PSW       ,GO BACK TO KERNEL MODE
3565  031356  020201                          CMP     R2,R1     ;DID PSW STAY IN USER, PRIORITY=0?
3566  031360  001401                          BEQ     4$        ,BRANCH IF YES
3567  031362  104060                          ERROR   60        ,PSW CHANGED BY AN RTI FROM USER
3568                                                            ,FOR A TIGHTER SCOPE LOOP
3569                                                            ,REPLACE ERROR CALL WITH
3570                                                            ,"BR=2$" = 000760
3571  031364  012737  031312  001110  4$      MOV     #1$,$LPERR ,RESET LOOP ON ERROR POINTER TO 1$
3572
3573
```

```
                                        ;;*****************************************************************
                                        ;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
                                        ;*
                                        ;;*****************************************************************
031372  000004                  TST44   SCOPE
031374  012737  000600  172350  1$      MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
031402  012705  077402                  MOV     #77402,R5       ;LOAD PDR4 DATA INTO R5
031406  010537  172310                  MOV     R5,KIPDR4       ;MAP PAGE 4 READ-ONLY
031412  012737  031442  000004          MOV     #4$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
031420  012737  031440  000250          MOV     #3$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
031426  012737  031434  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
031434  005237  060001          2$      INC     60001           ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
031440  104043                  3$      ERROR   43              ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000776
031442  012706  001100          4$      MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER TRAPPING
031446  005037  001200                  CLR     $TMP1           ;CLEAR ERROR INDICATOR
031452  013737  177572  001366          MOV     SR0,WASSR0      ;READ STATUS REG. 0
031460  013737  177576  001370  5$      MOV     SR2,WASSR2      ;READ STATUS REG. 2
031466  012700  000017                  MOV     #17,R0          ;LOAD EXPECTED SR0 CONTENTS  NTO R0
031472  020037  001366                  CMP     R0,WASSR0       ;SR0  ERROR BITS LEFT CLEAR BY TRAPPING?
031476  001402                          BEQ     6$              ;BRANCH IF YES
031500  005237  001200                  INC     $TMP1           ;SR0 ERROR BITS SET WHEN ODD ADDR  SERVICED
031504  012701  031460          6$      MOV     #5$,R1          ;LOAD EXPECTED SR2 CONTENTS INTO R1
031510  020137  001370                  CMP     R1,WASSR2       ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
031514  001402                          BEQ     7$              ;BRANCH IF YES
031516  005237  001200                  INC     $TMP1           ;SR2 LOCKED UP BY ODD ADDR  ERROR
031522  005737  001200          7$      TST     $TMP1           ;WHERE SR0 OR SR2 EFFECTED?
031526  001404                          BEQ     8$              ;BRANCH IF NO
031530  104044                          ERROR   44              ;SR0 OR SR2 CHANGED BY ODD ADDR  ERROR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000741
031532  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS THAT MAY BE SET IN SR0
031540  012737  002332  000004  8$      MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
031546  012737  002400  000250          MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M  TRAP HANDLER
031554  012737  077406  172310          MOV     #77406,KIPDR4   ;REMAP PAGE 4 TO READ/WRITE
031562  012737  031374  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
```

```
                                        T45     PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR  ERROR                SEQ 0074

                                        ;*      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
                                        ;*
                                        ;;**********************************************************************
031570  000004                  TST45   SCOPE
031572  004737  036002  177646  1$      JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
031576  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12-16K
031604  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12-16K
031612  012737  077402  177606          MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
031620  012737  077406  177610          MOV     #77406,UIPDR4   ;MAP USER PAGE 4 READ/WRITE
031626  012737  031702  000004          MOV     #4$,@#4         ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR ) VECTOR
031634  012737  140017  000006          MOV     #140017,@#6     ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
031642  012737  031702  000250          MOV     #4$,@#250       ;LOAD ADDRESS OF 4$ IN M.M  TRAP VECTOR
031650  012737  000340  000252          MOV     #340,@#252      ;LOAD A KERNEL PSW IN MMVEC+2
031656  012737  031664  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
031664  012737  140000  177776  2$      MOV     #140000,PSW     ;GO TO USER MODE
031672  012706  100002                  MOV     #100002,USP     ;SET USER STACK PTR  SO SECOND PUSH IS IN PG  3
031676  005237  100005          3$      INC     100005          ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
                                                                ;R/O ERROR WHEN TRY TO SAVE OLD PC
031702  016601  000002          4$      MOV     2(KSP),R1       ;PUT PSW SAVED ON KERNEL STACK INTO R1
031706  011603                          MOV     (KSP),R3        ;PUT PC SAVED ON KERNEL STACK INTO R3
031710  013737  177572  001366          MOV     SR0,WASSR0      ;READ THE CONTENTS OF M.M  STATUS REG  0
031716  013737  177576  001370          MOV     SR2,WASSR2      ;READ THE CONTENTS OF M.M.  STATUS REG  2
031724  042737  160000  177572          BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
031732  005037  177776                  CLR     PSW             ;BE SURE IN KERNEL MODE
031736  012706  001100                  MOV     #KERSTK,KSP     ;RESTORE KERNEL STACK POINTER
031742  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
031750  012706  000700                  MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
031754  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
031760  005037  001176                  CLR     $TMP0           ;CLEAR ERROR INDICATOR
031764  020127  170017                  CMP     R1,#170017      ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
                                                                ;ODD ADDR  TRAP FROM ERRVEC+2?
                                                                ;VALUE 170017 = PSW FROM LOC  6 WITH
                                                                ;PREVIOUS MODE BITS = USER
031770  001402                          BEQ     5$              ;BRANCH IF YES
031772  005237  001176                  INC     $TMP0           ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
```

```
3670  031776  020327  031702          5$      CMP     R3,#3$+4        ;WAS THE PC AT THE TIME OF THE ODD ADDR  ERROR
3671                                                                  ;SAVED ON THE STACK?
3672  032002  001402                          BEQ     6$              ;BRANCH IF YES
3673  032004  005237  001176                  INC     $TMPO           ;WRONG PC SAVED DURING TRAP SEQUENCE
3674  032010  023727  001366  020147  6$      CMP     WASSR0,#20147   ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
3675  032016  001402                          BEQ     7$              ;BRANCH IF YES
3676  032020  005237  001176                  INC     $TMPO           ;SR0 DID NOT REPORT R/O ABORT
3677  032024  023727  001370  031676  7$      CMP     WASSR2,#3$      ;DID SR2 LOCK UP VIRTUAL ADDR  OF LAST
3678                                                                  ;INSTRUCTION SUCCESSFULLY FETCHED?
3679  032032  001402                          BEQ     8$              ;BRANCH IF YES
3680  032034  005237  001176                  INC     $TMPO           ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR  INST
3681  032040  005737  001176          8$      TST     $TMPO           ;ANY "ERRORS" DURING TRAP SEQUENCE?
3682  032044  001401                          BEQ     9$              ;BRANCH IF NO
3683  032046  104045                          ERROR   45              ;THE WRONG PC OR PSW WERE SAVED
3684                                                                  ;OR SR0 OR SR2 DID NOT REPORT R/O
3685                                                                  ;ERROR DURING ODD ADDR. - KT TRAP
3686                                                                  ;SEQUENCE
3687                                                                  ;FOR TIGHTER SCOPE LOOP
3688                                                                  ;REPLACE ERROR CALL WITH
3689                                                                  ;"BR 2$" = 000710
3690  032050  012737  002332  000004  9$      MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3691  032056  012737  000340  000006          MOV     #340,@#6        ;RELOAD ERRVEC+2 WITH KERNEL PSW
3692  032064  012737  002400  000250          MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M.  TRAP HANDLER
3693  032072  012737  077406  177606          MOV     #77406,UIPDR3   ;REMAP USER PAGE 3 READ/WRITE
3694  032100  012737  031572  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3695  032106  004737  036036                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
3696                                ;;*****************************************************************
3697                                ;*
3698                                ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3699                                ;*      THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS
3700                                ;*
3701                                ;;*****************************************************************
3702
3703
3704                                ;;*****************************************************************
3705                                ;*TEST 46       MOVE FROM PREVIOUS (USER) I-SPACE
3706                                ;*
3707                                ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3708                                ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3709                                ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3710                                ;*
3711                                ;*
3712                                ;*      IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3713                                ;*      WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED
3714                                ;*
3715                                ;;*****************************************************************
3716  032112  000004                TST46:  SCOPE
3717  032114  005037  172340          1$      CLR     KIPAR0          ;MAP KERNEL PAGE 0 TO 0-4K
3718  032120  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL PAGE 1 TO 4-8K
3719  032126  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL PAGE 2 TO 8-12K
3720  032134  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL PAGE 3 TO 12-16K
3721  032142  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
3722  032150  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL PAGE 7 TO THE I/O PAGE
3723  032156  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3724                                                                  ;READ/WRITE, LENGTH 200 BLOCKS
3725  032162  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
```

```
3726  032166  012701  172300            MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3727  032172  010021            2$      MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3728  032174  077202                    SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL PDRS LOADED
3729  032176  012702  000010            MOV     #10,R2          ;SET LOOP COUNTER TO 8
3730  032202  012701  177600            MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3731  032206  010021            3$      MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3732  032210  077202                    SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL PDRS LOADED
3733  032212  012737  000000  177640    MOV     #000,UIPAR0     ;MAP USER I PAGE 0 TO 0-4K
3734  032220  012737  000200  177642    MOV     #200,UIPAR1     ;MAP USER I PAGE 1 TO 4-8K
3735  032226  012737  000400  177644    MOV     #400,UIPAR2     ;MAP USER I PAGE 2 TO 8-12K
3736  032234  012737  000600  177646    MOV     #600,UIPAR3     ;MAP USER I PAGE 3 TO 12-16K
3737  032242  012737  007600  177656    MOV     #7600,UIPAR7    ;MAP USER I PAGE 7 TO THE I/O PAGE
3738  032250  012737  032256  001110    MOV     #4$,$LPERR      ;SET LOOP ON ERROR TO 4$
3739  032256                    4$
3740  032256  012737  077406  172310    MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3741  032264  012737  000600  172350    MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3742  032272  012737  000600  177650    MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3743  032300  012700  036514            MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3744  032304  010037  100000            MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3745  032310  012737  032712  000250    MOV     #23$,MMVEC      ;SET M.M. VECTOR TO 23$
3746  032316  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3747                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3748                                    ;
3749  032322  012737  032330  001110    MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3750  032330  012737  030340  177776  5$ MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3751  032336  006506            6$      MFPI    USP             ;PUT USER STACK POINTER ON KERNEL
3752                                                            ;STACK
3753  032340  022706  001100            CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
```

```
3754  032344  001407                    BEQ     7$              ;BRANCH IF NOTHING WAS PUSHED
3755  032346  012600                    MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3756  032350  012701  000700            MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
```

```
3757  032354  020001                        CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3758  032356  001403                        BEQ     8$              ;BRANCH IF YOU DID
3759  032360  104046                        ERROR   46              ;WRONG THING WAS PUSHED ON STACK
3760                                                                ;FOR TIGHTER SCOPE LOOP
3761                                                                ;REPLACE ERROR CALL WITH
3762                                                                ;"BR 5$" = 000763
3763  032362  000401                        BR      8$              ;BRANCH TO NEXT TRY
3764  032364  104050                7$      ERROR   50              ;NOTHING PUSHED ON STACK
3765                                                                ;FOR TIGHTER SCOPE LOOP
3766                                                                ;REPLACE ERROR CALL WITH
3767                                                                ;"BR 5$" = 000761
3768  032366                        8$      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3769  032366  012737  032400  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3770  032374  012700  036514                MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3771  032400  012737  030340  177776  9$    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3772  032406  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3773  032412  006512                        MFPI    (R2)            ;READ FROM PHYSICAL 60000
3774  032414  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3775  032416  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3776  032420  001401                        BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3777  032422  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3778                                                                ;FOR TIGHTER SCOPE LOOP
3779                                                                ;REPLACE ERROR CALL WITH
3780                                                                ;"BR 9$" = 000766
3781  032424                        10$.    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3782  032424  012737  032432  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3783  032432  012737  030340  177776  11$.  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3784  032440  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3785  032444  006522                        MFPI    (R2)+           ;READ FROM PHYSICAL 60000
?     032446  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
37.     ..450 020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3788  0..452  001401                        BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3789  032454  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3790                                                                ;FOR TIGHTER SCOPE LOOP
3791                                                                ;REPLACE ERROR CALL WITH
3792                                                                ;"BR 11$" = 000766
3793  032456                        12$     ;THE FOLLOWING WILL TEST DSTM=3 MFPI
```

```
3794   032456  012737  032464  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3795   032464  012737  030340  177776  13$     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3796   032472  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
3797   032476  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3798   032500  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3799   032502  001401                          BEQ     14$             ;BRANCH IF CORRECT DATA WAS FETCHED
3800   032504  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3801                                                                   ;FOR TIGHTER SCOPE LOOP
3802                                                                   ;REPLACE ERROR CALL WITH
```

```
                                                        ,"BR 13$" = 000767
032506                                  14$     ,THE FOLLOWING WILL TEST DSTM=4 MFPI
032506  012737  032514  001110                MOV     #15$,$LPERR     ,SET LOOP ON ERROR POINTER TO 15$
032514  012737  030340  177776  15$           MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
032522  012702  100002                        MOV     #100002,R2      ,LOAD VIRTUAL ADDRESS INTO R2
032526  006542                                MFPI    -(R2)           ,READ FROM PHYSICAL 60000
032530  012601                                MOV     (KSP)+,R1       ,POP KERNEL STACK INTO R1
032532  020001                                CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
032534  001401                                BEQ     16$             ,BRANCH IF CORRECT DATA WAS FETCHED
032536  104046                                ERROR   46              ,WRONG DATA WAS FETCHED
                                                                      ,FOR TIGHTER SCOPE LOOP
                                                                      ,REPLACE ERROR CALL WITH
                                                                      ,"BR 15$" = 000766
032540                                  16$
                                                ,THE FOLLOWING WILL TEST DSTM=5 MFPI
                                                ,
032540  012737  032546  001110                MOV     #17$,$LPERR     ,SET LOOP ON ERROR POINTER TO 17$
032546  012737  030340  177776  17$           MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
032564  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT  ADDR INTO LOC  $TMP2
032562  012702  001204                        MOV     #($TMP2+2),R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
032566  006552                                MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
032570  012601                                MOV     (KSP)+,R1       ,POP KERNEL STACK INTO R1
032572  020001                                CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
032574  001401                                BEQ     18$             ;BRANCH IF CORRECT DATA WAS FETCHED
032576  104046                                ERROR   46              ;WRONG DATA WAS FETCHED
                                                                      ,FOR TIGHTER SCOPE LOOP
                                                                      ,REPLACE ERROR CALL WITH
                                                                      ,"BR 17$" = 000763
032600                                  18$     ,THE FOLLOWING WILL TEST DSTM=6 MFPI
                                                ,
032600  012737  032606  001110                MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
032606  012737  030340  177776  19$.          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
032614  005002                                CLR     R2              ;MAKE REGISTER 2 A ZERO
032616  006562  100000                        MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
032622  012601                                MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
032624  020001                                CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
032626  001401                                BEQ     20$             ,BRANCH IF CORRECT DATA WAS FETCHED
032630  104046                                ERROR   46              ,WRONG DATA WAS FETCHED
                                                                      ,FOR TIGHTER SCOPE LOOP
                                                                      ;REPLACE ERROR CALL WITH
                                                                      ,"BR 19$" = 000766
032632                                  20$     ,THE FOLLOWING WILL TEST DSTM=7 MFPI
                                                ,
032632  012737  032640  001110                MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
032640  012737  030340  177776  21$           MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
032646  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A.  INTO $TMP2
032664  012702  001202                        MOV     #$TMP2,R2       ,LOAD ADDRESS OF $TMP2 INTO R2
032660  006572  000000                        MFPI    @0(R2)          ,USE $TMP2 TO FETCH VIRTUAL
                                                                      ,ADDRESS OF 60000
032664  012601                                MOV     (KSP)+,R1       ,POP KERNEL STACK INTO R1
032666  020001                                CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
032670  001401                                BEQ     22$             ,BRANCH IF CORRECT DATA WAS FETCHED
032672  104046                                ERROR   46              ,WRONG DATA WAS FETCHED
                                                                      ,FOR TIGHTER SCOPE LOOP
                                                                      ,REPLACE ERROR CALL WITH
                                                                      ,"BR 21$" = 000762
```

```
3859  032674  012737  002400  000250  22$     MOV     #MGMERR,MMVEC   ,SET M M  VECTOR TO NORMAL ROUTINE
3860  032702  012737  032114  001110          MOV     #1$,$LPERR      ,SET LOOP POINTER TO START OF TEST
3861  032710  000423                          BR      TST47           ,,BRANCH TO NEXT TEST
3862
3863
3864  032712  012637  001356          23$     MOV     (KSP)+,TRAPPC   ,SAVE PC & PS OF TRAP
3865  032716  012637  001360                  MOV     (KSP)+,TRAPPS
3866  032722  013737  177572  001366          MOV     SR0,WASSR0      ,SAVE SR0 FOR ERROR TYPEOUT
3867  032730  013737  177576  001370          MOV     SR2,WASSR2      ,SAVE SR2 FOR ERROR TYPEOUT
3868  032736  042737  160000  177572          BIC     #160000,SR0     ,CLEAR ERROR BITS IN SR0 AND LEAVE
3869  032744  104051                          ERROR   51              ,TRIED TO READ NON-RESIDENT PAGE
3870                                                                  ,FOR TIGHTER SCOPE LOOP
3871                                                                  ,REPLACE ERROR CALL WITH
3872                                                                  ,A "NOP" = 000240
3873  032746  013746  001360                  MOV     TRAPPS,-(KSP)   ,PUT PC & PS OF TRAP ON STACK
3874  032752  013746  001356                  MOV     TRAPPC,-(KSP)
3875  032756  000002                          RTI
3876
3877
3878                                  ,,**********************************************************************
3879                                  ,*TEST 47       MOVE TO PREVIOUS (USER) I-SPACE
3880                                  ,*
3881                                  ,*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3882                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3883                                  ,*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3884                                  ,*
3885                                  ,*
3886                                  ,*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3887                                  ,*      WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED
3888                                  ,*
3889                                  ,;,,**********************************************************************
3890  032760  000004          TST47.  SCOPE
3891  032762  012737  077406  172310  1$      MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3892  032770  012737  077406  177610          MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
3893  032776  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3894  033004  012737  000600  177650          MOV     #600,UIPAR4     ,MAP USER I PAGE 4 TO 12K
3895  033012  012737  033572  000250          MOV     #20$,MMVEC      ,SET M.M.  VECTOR TO 20$
3896                                          ,THE FOLLOWING WILL TEST DSTM=0 MTPI
3897                                          ,
3898  033020  012737  030340  177776  2$      MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
3899  033026  012746  007777                  MOV     #7777,-(KSP)    ,PUSH DATA ON KERNEL STACK
3900  033032  006606                          MTPI    USP             ;LOAD USER STACK POINTER
3901  033034  006506                          MFPI    USP             ;READ USER STACK POINTER
3902  033036  012601                          MOV     (KSP)+,R1       ,POP KERNEL STACK INTO R1
3903  033040  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
3904  033044  001401                          BEQ     3$              ;BRANCH IF IT WAS
3905  033046  104050                          ERROR   50              ;USER STACK POINTER NOT CHANGED
3906                                                                  ,FOR TIGHTER SCOPE LOOP
3907                                                                  ;REPLACE ERROR CALL WITH
3908                                                                  ,"BR 2$" = 000764
3909  033050  012737  030340  177776  3$      MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
3910  033056  012746  000700                  MOV     #USESTK,-(KSP)  ,GET READY TO RESTORE USER S  POINT
3911  033062  006606                          MTPI    USP             ,RESTORE USER STACK POINTER
3912  033064                          4$      ,THIS WILL TEST DSTM = 1 MTPI
3913  033064  012737  033102  001110          MOV     #5$,$LPERR      ,SET LOOP ON ERROR POINTER TO 5$
3914  033072  012702  100000                  MOV     #100000,R2      ,LOAD VIRTUAL ADDRESS INTO R2
```

```
3915  033076  012700  125252            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3916  033102  010046                5$   MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3917  033104  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3918  033110  006612                      MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3919  033112  112737  000006  172310     MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3920  033120  011201                      MOV     (R2),R1         ;READ FROM ADDRESS 60000
3921  033122  020001                      CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
3922  033124  001401                      BEQ     6$              ;BRANCH IF STORE WAS CORRECT
3923  033126  104047                      ERROR   47              ;INCORRECT STORE
3924                                                            ;FOR TIGHTER SCOPE LOOP
3925                                                            ;REPLACE ERROR CALL WITH
3926                                                            ;"BR 5$" = 000765
3927  033130                        6$    ;THE FOLLOWING WILL TEST OSTM=2 MTPI.
3928                                      ;
3929  033130  012737  033154  001110     MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
3930  033136  012737  030340  177776     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3931  033144  012700  125252            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3932  033150  012702  100000            MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3933  033154  010046                8$   MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3934  033156  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3935  033162  006612                      MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3936  033164  112737  000006  172310     MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESISENT
3937  033172  013701  100000            MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3938  033176  020001                      CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3939  033200  001401                      BEQ     9$              ;BRANCH IF STORE WAS CORRECT
3940  033202  104047                      ERROR   47              ;INCORRECT STORE
3941                                                            ;FOR TIGHTER SCOPE LOOP
3942                                                            ;REPLACE ERROR CALL WITH
3943                                                            ;"BR 8$" = 000764
3944  033204                        9$    ;THIS WILL TEST OSTM = 3 MTPI.
3945  033204  012737  033224  001110     MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3946  033212  012737  030340  177776     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USEP
3947  033220  012700  052525            MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3948  033224  010046                10$  MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3949  033226  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3950  033232  006637  100000            MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
3951  033236  112737  000006  172310     MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3952  033244  013701  100000            MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3953  033250  020001                      CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
```

```
3954  033252  001401                        BEQ     11$             ,BRANCH IF STORE WAS CORRECT
3955  033254  104047                        ERROR   47              ,INCORRECT STORE
3956                                                                  ,FOR TIGHTER SCOPE LOOP
3957                                                                  ,REPLACE ERROR CALL WITH
3958                                                                  ,"BR 10$" = 000763
3959  033256                          11$     ,THIS WILL TEST DSTM = 4 MTPI
3960  033256  012737  033276  001110        MOV     #12$,$LPERR     ,SET LOOP ON ERROR POINTER TO 12$
3961  033264  012737  030340  177776        MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
3962  033272  012700  125252                MOV     #125252,R0      ,LOAD TEST DATA INTO R0
3963  033276  010046                  12$   MOV     R0,-(KSP)       ,PUSH TEST DATA ON KERNEL STACK
3964  033300  012702  100002                MOV     #100002,R2      ,LOAD VIRTUAL ADDRESS INTO R2
```

```
3965  033304  105037  172310          CLRB    KIPDR4          ,MAKE KERNEL I PAGE 4 NON-RESIDENT
3966  033310  006642                  MTPI    -(R2)           ,LOAD TEST DATA INTO PHYSICAL 60000
3967  033312  112737  000006  172310  MOVB    #006,KIPDR4     ,MAKE KERNEL PAGE 4 RESIDENT
3968  033320  013701  100000          MOV     @#100000,R1     ,READ FROM ADDRESS 60000
3969  033324  020001                  CMP     R0,R1           ,SEE IF DATA WAS STORED CORRECTLY
3970  033326  001401                  BEQ     13$             ,BRANCH IF STORE WAS CORRECT
3971  033330  104047                  ERROR   47              ,INCORRECT STORE
3972                                                          ,FOR TIGHTER SCOPE LOOP
3973                                                          ,REPLACE ERROR CALL WITH
3974                                                          ,"BR 12$" = 000762
3975  033332                  13$     ,THE FOLLOWING WILL TEST DSTM=5 MTPI
3976                                  ,
3977  033332  012737  033364  001110  MOV     #14$,$LPERR     ,SET LOOP ON ERROR POINTER TO 14$
3978  033340  012737  030340  177776  MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
3979  033346  012700  052525          MOV     #52525,R0       ,LOAD TEST DATA INTO R0
3980  033352  012702  001204          MOV     #($TMP2+2),R2   ,LOAD ADDR. OF LOC. $TMP2+2  NTO R2
3981  033356  012737  100000  001202  MOV     #100000,$TMP2   ,LOAD VIRT. ADDR. OF TEST LOC  INTO $TMP2
3982  033364  010046          14$     MOV     R0,-(KSP)       ,PUSH TEST DATA ON KERNEL STACK
3983  033366  105037  172310          CLRB    KIPDR4          ,MAKE KERNEL PAGE 4 NON-RESIDENT
3984  033372  006652                  MTPI    @-(R2)          ,LOAD TEST DATA INTO PHYSICAL 60000
3985  033374  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3986  033402  013701  100000          MOV     @#100000,R1     ,READ FROM ADDRESS 60000
3987  033406  020001                  CMP     R0,R1           ,SEE IF DATA WAS STORED CORRECTLY
3988  033410  001401                  BEQ     15$             ,BRANCH IF STORE WAS CORRECT
3989  033412  104047                  ERROR   47              ,INCORRECT STORE
3990                                                          ,FOR TIGHTER SCOPE LOOP
3991                                                          ,REPLACE ERROR CALL WITH
3992                                                          ,"BR 14$" = 000764
3993  033414                  15$     ,THIS WILL TEST DSTM = 6 MTPI
3994                                  ,
3995  033414  012737  033436  001110  MOV     #16$,$LPERR     ,SET LOOP ON ERROR POINTER TO 16$
3996  033422  012737  030340  177776  MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
3997  033430  012700  052525          MOV     #52525,R0       ,LOAD TEST DATA INTO R0
3998  033434  005002                  CLR     R2              ,MAKE REGISTER 2 ZERO
3999  033436  010046          16$     MOV     R0,-(KSP)       ,PUSH TEST DATA ON KERNEL STACK
4000  033440  105037  172310          CLRB    KIPDR4          ,MAKE KERNEL I PAGE 4 NON-RESIDENT
4001  033444  006662  100000          MTPI    100000(R2)      ,LOAD TEST DATA INTO PHYSICAL 60000
4002  033450  112737  000006  172310  MOVB    #006,KIPDR4     ,MAKE KERNEL PAGE 4 RESIDENT
4003  033456  013701  100000          MOV     @#100000,R1     ,READ FROM ADDRESS 60000
4004  033462  020001                  CMP     R0,R1           ,SEE IF DATA WAS STORED CORRECTLY
4005  033464  001401                  BEQ     17$             ,BRANCH IF STORE WAS CORRECT
4006  033466  104047                  ERROR   47              ,INCORRECT STORE
4007                                                          ,FOR TIGHTER SCOPE LOOP
4008                                                          ,REPLACE ERROR CALL WITH
4009                                                          ,"BR 16$" = 000763
4010  033470                  17$     ,THE FOLLOWING WILL TEST DSTM=7 MTPI
4011                                  ,
4012  033470  012737  033522  001110  MOV     #18$,$LPERR     ,SET LOOP ON ERROR POINTER TO 18$
4013  033476  012737  030340  177776  MOV     #030340,PSW     ,MAKE PREVIOUS MODE USER
4014  033504  012700  125252          MOV     #125252,R0      ,LOAD TEST DATA INTO R0
4015  033510  012737  100000  001202  MOV     #100000,$TMP2   ,LOAD VIRT  ADDR. OF TEST LOCATION
4016                                                          ,INTO LOCATION $TMP2
4017  033516  012702  001202          MOV     #$TMP2,P2       ,LOAD ADDRESS OF $TMP2 INTO R2
4018  033522  010046          18$     MOV     R0,-(KSP)       ,PUSH TEST DATA ON KERNEL STACK
4019  033524  105037  172310          CLRB    KIPDR4          ,MAKE KERNEL PAGE 4 NON-RESIDENT
4020  033530  006672  000000          MTPI    @0(R2)          ,LOAD TEST DATA INTO PHYSICAL 60000
```

```
4021  033534  112737  000006  172310          MOVB    #006,KIPDR4     ,MAKE KERNEL PAGE 4 RESIDENT
4022  033542  013701  100000                  MOV     @#100000,R1     ,READ FROM ADDRESS 60000
4023  033546  020001                          CMP     R0,R1           ,SEE IF DATA WAS STORED CORRECTLY
4024  033550  001401                          BEQ     19$             ,BRANCH IF STORE WAS CORRECT
4025  033552  104047                          ERROR   47              ,INCORRECT STORE
4026                                                                  ,FOR TIGHTER SCOPE LOOP
4027                                                                  ,REPLACE ERROR CALL WITH
4028                                                                  ,"BR 18$" = 000763
4029  033554  012737  032762  001110  19$     MOV     #1$,$LPERR      ,SET LOOP POINTER TO START OF TEST
4030  033562  012737  002400  000250          MOV     #MGMERR,MMVEC   ,RESTORE M.M. VECTOR TO NORMAL ROUTINE
4031  033570  000423                          BR      TST50           ,,BRANCH TO NEXT TEST
4032
4033
4034  033572  012637  001356          20$     MOV     (KSP)+,TRAPPC   ,SAVE PC & PS OF TRAP
4035  033576  012637  001360                  MOV     (KSP)+,TRAPPS
4036  033602  013737  177572  001366          MOV     SR0,WASSR0      ,SAVE SR0 FOR ERROR TYPEOUT
4037  033610  013737  177576  001370          MOV     SR2,WASSR2      ,SAVE SR2 FOR ERROR TYPEOUT
4038  033616  042737  160000  177572          BIC     #160000,SR0     ,CLEAR ERROR BITS IN SR0
4039  033624  104051                          ERROR   51              ,TRIED TO LOAD A N R  PAGE 4
4040                                                                  ,FOR TIGHTER SCOPE LOOP
4041                                                                  ;REPLACE ERROR CALL WITH
4042                                                                  ,A "NOP" = 000240
4043  033626  013746  001360                  MOV     TRAPPS,-(KSP)   ,PUT PC & PS OF TRAP ON STACK
4044  033632  013746  001356                  MOV     TRAPPC -(KSP)
4045  033636  000002                          RTI                     ,RETURN TO TEST
4046
4047
4048
4049                                  ,,**********************************************************************
4050                                  ,*TEST 50       MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
4051                                  ,*
4052                                  ,*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
4053                                  ,*      FETCH IS FROM KERNEL SPACE
4054                                  ,*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
4055                                  ,*
4056                                  ,*
4057                                  ,*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
4058                                  ,*      WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED
4059                                  ,*
4060                                  ,,**********************************************************************
4061  033640  000004                  TST50   SCOPE
4062  033642  012700  077406          1$      MOV     #77406,R0       ,MAKE ALL USER I-SPACE PAGES RESIDENT
4063                                                                  ,READ/WRITE, LENGTH 200 BLOCKS
4064  033646  012702  000010                  MOV     #10,R2          ,SET LOOP COUNTER TO 8
4065  033652  012701  177600                  MOV     #UIPDR0,R1      ,LOAD ADDRESS OF FIRST PDR IN R1
4066  033656  010021                  2$      MOV     R0,(R1)+        ,LOAD PDR WITH 77406
4067  033660  077202                          SOB     R2,2$           ,LOOP UNTIL 8 USER PDRS LOADED
4068  033662  012737  033670  001110          MOV     #3$,$LPERR      ,SET LOOP ON ERROR TO 3$
4069  033670  012737  140340  177776  3$      MOV     #140340,PSW     ,GO TO USER MODE FOR THIS TEST
4070  033676  012737  077406  172310          MOV     #77406,KIPDR4   ,KERNEL I-SPACE PAGE 4 READ/WRITE
4071  033704  012737  000600  172350          MOV     #600,KIPAR4     ,MAP KERNEL I PAGE 4 TO 12K
4072  033712  012737  000600  177650          MOV     #600,UIPAR4     ,MAP USER I PAGE 4 TO 12K
4073  033720  012700  036514                  MOV     #36514,R0       ,LOAD DATA PATTERN INTO R0
4074  033724  010037  100000                  MOV     R0,@#100000     ,LOAD DATA PATTERN INTO PHY 60000
4075  033730  012702  100000                  MOV     #100000,R2      ,LOAD VIRTUAL ADDRESS INTO R2
4076                                          ,THE FOLLOWING WILL TEST DSTM=0 MFPI
```

```
4077
4078  033734  012737  034336  000250          MOV     #21$,MMVEC      ;SET M M  VECTOR TO 21$
4079  033742  105037  177610                  CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
4080  033746  012737  140340  177776          MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4081  033754  006506                  4$      MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
4082  033756  022706  000700                  CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT 1$
4083  033762  001407                          BEQ     5$              ;BRANCH IF NOTHING WAS PUSHED
4084  033764  012600                          MOV     (USP)+,R0       ;POP USER STACK INTO R0
4085  033766  012701  001100                  MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
4086  033772  020001                          CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
4087  033774  001403                          BEQ     6$              ;BRANCH IF YOU DID
4088  033776  104046                          ERROR   46              ;WRONG THING WAS PUSHED ON STACK
4089                                                                  ;FOR TIGHTER SCOPE LOOP
4090                                                                  ;REPLACE ERROR CALL WITH
4091                                                                  ;"BR 4$" = 000766
4092  034000  000401                          BR      6$              ;BRANCH TO NEXT TRY
4093  034002  104050                  5$      ERROR   50              ;NOTHING PUSHED ON STACK
4094                                                                  ;FOR TIGHTER SCOPE LOOP
4095                                                                  ;REPLACE ERROR CALL WITH
4096                                                                  ;"BR 4$" = 000764
4097  034004                          6$      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
4098  034004  012737  034012  001110          MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
4099  034012  012737  140340  177776  7$      MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4100  034020  012700  036514                  MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
4101  034024  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4102  034030  006512                          MFPI    (R2)            ;READ FROM PHYSICAL 60000
4103  034032  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
4104  034034  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4105  034036  001401                          BEQ     8$              ;BRANCH IF CORRECT DATA WAS FETCHED
4106  034040  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
4107                                                                  ;FOR TIGHTER SCOPE LOOP
4108                                                                  ;REPLACE ERROR CALL WITH
4109                                                                  ;"BR 7$" = 000764
4110  034042                          8$      ;THE FOLLOWING WILL TEST DSM=2 MFPI.
4111  034042  012737  034050  001110          MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
4112  034050  012737  140340  177776  9$      MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4113  034056  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4114  034062  006522                          MFPI    (R2)+           ;READ FROM PHYSICAL 60000
4115  034064  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
4116  034066  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4117  034070  001401                          BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
4118  034072  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
4119                                                                  ;FOR TIGHTER SCOPE LOOP
4120                                                                  ;REPLACE ERROR CALL WITH
4121                                                                  ;"BR 9$" = 000766
4122  034074                          10$     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
4123  034074  012737  034102  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
4124  034102  012737  140340  177776  11$     MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4125  034110  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
4126  034114  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
4127  034116  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4128  034120  001401                          BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
4129  034122  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
4130                                                                  ;FOR TIGHTER SCOPE LOOP
4131                                                                  ;REPLACE ERROR CALL WITH
4132                                                                  ;"BR 11$" = 000767
```

```
4133  034124                                  12$     ,THE FOLLOWING WILL TEST DSTM=4 MFPI
4134  034124  012737  034132  001110                  MOV     #13$,$LPERR     ,SET LOOP ON ERROR POINTER TO 13$
4135  034132  012737  140340  177776  13$     MOV     #140340,PSW     ,MAKE PREVIOUS MODE KERNEL PRESENT USER
4136  034140  012702  100002                  MOV     #100002,R2      ,LOAD VIRTUAL ADDRESS INTO R2
4137  034144  006542                          MFPI    -(R2)           ,READ FROM PHYSICAL 60000
4138  034146  012601                          MOV     (USP)+,R1       ,POP USER STACK INTO R1
4139  034150  020001                          CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
4140  034152  001401                          BEQ     14$             ,BRANCH IF CORRECT DATA WAS FETCHED
4141  034154  104046                          ERROR   46              ,WRONG DATA WAS FETCHED
4142                                                                  ,FOR TIGHTER SCOPE LOOP
4143                                                                  ,REPLACE ERROR CALL WITH
4144                                                                  ,"BR 13$" = 000766
4145  034156                                  14$     ,THE FOLLOWING WILL TEST DSTM=5 MFPI.
4146                                                  ,
4147  034156  012737  034164  001110                  MOV     #15$,$LPERR     ,SET LOOP ON ERROR POINTER TO 15$
4148  034164  012737  140340  177776  15$     MOV     #140340,PSW     ,MAKE PREVIOUS MODE KERNEL PRESENT USER
4149  034172  012737  100000  001202                  MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC  $TMP2
4150  034200  012702  001204                  MOV     #<$TMP2+2>,R2   ,LOAD ADDRESS OF $TMP2+2 INTO R2
4151  034204  006552                          MFPI    @-(R2)          ,READ FROM PHYSICAL 60000
4152  034206  012601                          MOV     (USP)+,R1       ,POP USER STACK INTO R1
4153  034210  020001                          CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
4154  034212  001401                          BEQ     16$             ,BRANCH IF CORRECT DATA FETCHED
4155  034214  104046                          ERROR   46              ,WRONG DATA WAS FETCHED
4156                                                                  ;FOR TIGHTER SCOPE LOOP
4157                                                                  ;REPLACE ERROR CALL WITH
4158                                                                  ,"BR 15$" = 000763
4159  034216                                  16$     ,THE FOLLOWING WILL TEST DSTM=6 MFPI.
4160                                                  ,
4161  034216  012737  034224  001110                  MOV     #17$,$LPERR     ,SET LOOP ON ERROR POINTER TO 17$
4162  034224  012737  140340  177776  17$     MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4163  034232  005002                          CLR     R2              ,MAKE REGISTER 2 A ZERO
4164  034234  006562  100000                  MFPI    100000(R2)      ,READ FROM PHYSICAL 60000
4165  034240  012601                          MOV     (USP)+,R1       ,POP USER STACK INTO R1
4166  034242  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
4167  034244  001401                          BEQ     18$             ;BRANCH IF CORRECT DATA FETCHED
4168  034246  104046                          ERROR   46              ,WRONG DATA WAS FETCHED
4169                                                                  ,FOR TIGHTER SCOPE LOOP
4170                                                                  ,REPLACE ERROR CALL WITH
4171                                                                  ,"BR 17$" = 000766
4172  034250                                  18$     ,THE FOLLOWING WILL TEST DSTM=7 MFPI
4173                                                  ,
4174  034250  012737  034256  001110                  MOV     #19$,$LPERR     ,SET LOOP ON ERROR POINTER TO 19$
4175  034256  012737  140340  177776  19$     MOV     #140340,PSW     ,MAKE PREVIOUS MODE KERNEL PRESENT USER
4176  034264  012737  100000  001202                  MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR  INTO $TMP2
4177  034272  012702  001202                  MOV     #$TMP2,R2       ,LOAD ADDRESS OF $TMP2 INTO R2
4178  034276  006572  000000                  MFPI    @0(R2)          ,READ FROM PHYSICAL 60000
4179  034302  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
4180  034304  020001                          CMP     R0,R1           ,WAS DATA FETCHED SAME AS STORED
4181  034306  001401                          BEQ     20$             ,BRANCH IF CORRECT DATA FETCHED
4182  034310  104046                          ERROR   46              ,WRONG DATA WAS FETCHED
4183                                                                  ,FOR TIGHTER SCOPE LOOP
4184                                                                  ,REPLACE ERROR CALL WITH
4185                                                                  ,"BR 19$" = 000762
4186  034312  012737  002400  000250  20$     MOV     #MGMERR,MMVEC   ,SET M M  VECTOR TO NORMAL ROUTINE
4187  034320  012737  000340  177776          MOV     #00340,PSW      ,GO BACK TO KERNEL MODE, PREVIOUS KERNEL
4188  034326  012737  033642  001110          MOV     #1$,$LPERR      ,SET LOOP POINTER TO START OF TEST
```

```
 4189  034334  000423                          BR      TST51           ;;BRANCH TO NEXT TEXT
 4190
 4191
 4192  034336  012637  001356          21$     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
 4193  034342  012637  001360                  MOV     (KSP)+,TRAPPS
 4194  034346  013737  177572  001366          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
 4195  034354  013737  177576  001370          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
 4196  034362  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
 4197  034370  104051                          ERROR   51              ;TRIED TO READ NON-RESIDENT PAGE
 4198                                                                  ;FOR TIGHTER SCOPE LOOP
 4199                                                                  ;REPLACE ERROR CALL WITH
 4200                                                                  ;A "NOP" = 000240
 4201  034372  013746  001360                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
 4202  034376  013746  001356                  MOV     TRAPPC,-(KSP)
 4203  034402  000002                          RTI                     ;RETURN TO TEST
 4204
 4205                                  ;;*****************************************************************
 4206                                  ;*TEST 51        MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
 4207                                  ;*
 4208                                  ;*      THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
 4209                                  ;*      BETWEEN INSTRUCTION AND DATA SPACE IN THE 11/34
 4210                                  ;*      MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI
 4211                                  ;*
 4212                                  ;;*****************************************************************
 4213  034404  000004                  TST51:  SCOPE
 4214  034406  012737  030340  177776  1$      MOV     #030340,PSW     ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
 4215  034414  106506                          MFPD    USP             ;MFPD SHOULD ACT LIKE MFPI PUTTING
 4216                                                                  ;USER STACK POINTER ON THE KERNEL STACK
 4217  034416  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON KERNEL STACK?
 4218  034422  001407                          BEQ     2$              ;BRANCH IF NO
 4219  034424  012600                          MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
 4220  034426  012701  000700                  MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
 4221  034432  020001                          CMP     R0,R1           ;DID GET RIGHT POINTER VALUE?
 4222  034434  001403                          BEQ     3$              ;BRANCH IF YES
 4223  034436  104053                          ERROR   53              ;WRONG THING WAS PUSHED ON STACK
 4224                                                                  ;FOR TIGHTER SCOPE LOOP
 4225                                                                  ;REPLACE ERROR CALL WITH
 4226                                                                  ;"BR 1$" = 000763
 4227  034440  000401                          BR      3$              ;BRANCH TO NEXT TRY
 4228  034442  104054                  2$      ERROR   54              ;NOTHING PUSHED ON STACK
 4229                                                                  ;FOR TIGHTER SCOPE LOOP
 4230                                                                  ;REPLACE ERROR CALL WITH
 4231                                                                  ;"BR 1$" = 000761
 4232  034444  012737  034452  001110  3$      MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
 4233  034452  012746  007777          4$      MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
 4234  034456  106606                          MTPD    USP             ;LOAD THE USER STACK POINTER
 4235  034460  106506                          MFPD    USP             ;READ USER STACK POINTER
 4236  034462  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
 4237  034464  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED?
 4238  034470  001401                          BEQ     5$              ;BRANCH IF YES
 4239  034472  104054                          ERROR   54              ;USER STACK POINTER NOT CHANGED
 4240                                                                  ;FOR TIGHTER SCOPE LOOP
 4241                                                                  ;REPLACE ERROR CALL WITH
 4242                                                                  ;"BR 4$" = 000767
 4243  034474  012746  000700          5$      MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK  PTR
 4244  034500  106606                          MTPD    USP             ;RESTORE USER STACK POINTER
```

```
 4245  034502  012737  034406  001110          MOV     #1$,SLPERR      ;SET LOOP POINTER TO START OF TEST
 4246
 4247                                  ;;*****************************************************************
 4248                                  ;*TEST 52       MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
 4249                                  ;*
 4250                                  ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
 4251                                  ;*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
 4252                                  ;*      STACK IS NOT DECREMENTED BEFORE ACCESS.
 4253                                  ;*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
 4254                                  ;*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
 4255                                  ;;*****************************************************************
 4256  034510  000004                 TST52   SCOPE
 4257  034512  005037  177776          1$      CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
 4258  034516  012700  001100                  MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
 4259  034522  010006                          MOV     R0,KSP          ;LOAD STACK POINTER
 4260  034524  006506                          MFPI    KSP             ;THE VALUE "STACK" SHOULD BE PUSHED
 4261                                                                  ;BEFORE BEING DECREMENTED
 4262  034526  011601                          MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
 4263  034530  02000.                          CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE
 4264                                                                  ;STACK POINTER PUSHED?
 4265  034532  001401                          BEQ     2$              ;BRANCH IF YES
 4266  034534  104046                          ERROR   46              ;MFPI FETCHED WRONG DATA
 4267                                                                  ;FOR TIGHTER SCOPE LOOP
 4268                                                                  ;REPLACE ERROR CALL WITH
 4269                                                                  ;"BR 1$" = 000766
 4270  034536  005740                  2$      TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
 4271  034540  020600                          CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
 4272  034542  001401                          BEQ     3$              ;BRANCH IF YES
 4273  034544  104050                          ERROR   50              ;STACK NOT PUSHED BY THE MFPI
 4274                                                                  ;FOR TIGHTER SCOPE LOOP
 4275                                                                  ;REPLACE ERROR CALL WITH
 4276                                                                  ;"BR 1$" = 000762
 4277  034546  012706  001100          3$      MOV     #STACK,KSP      ;RESTORE STACK POINTER
 4278
 4279
 4280
 4281
 4282
 4283                                  SBTTL   ;*****************************************************************
 4284
```

```
                                        .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

034552                                  $EOP
034552  000004                                  SCOPE
034554  005037  001102                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
034560  005037  001212                          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
034564  005237  001234                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
034570  042737  100000  001234                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG  NUMBER
034576  005327                                  DEC     (PC)+           ;;LOOP?
034600  000001                          $EOPCT.  WORD   1
034602  003072                                  BGT     $DOAGN          ;;YES
034604  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
034606  000001                          $ENDCT.  WORD   1
034610  034600                                  $EOPCT
034612  104401  034620                          TYPE    ,65$            ;;TYPE ASCIZ STRING
034616  000407                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$    ASCIZ  <12><15>/END PASS #/
034636                                  64$
034636  013746  001234                          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
034642  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
034644  104401  034652                          TYPE    ,67$            ;;TYPE ASCIZ STRING
034650  000421                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$.   .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
034714                                  66$
034714  013746  0J1112                          MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
034720  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
034722  104401  001223                          TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
034726  005037  001112                          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
034732  013700  000042                  $GET42. MOV     @#42,R0         ;;GET MONITOR ADDRESS
034736  001414                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
034740  005046                                  CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
034742  012746  034750                          MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
034746  000426                                  BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                        ;;WITH A CLEARED "T" BIT
034750                                  $CLR.T:
034750  013700  000042                          MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
034754  001405                                  BEQ     $DOAGN          ;;RETURN ADDRESS
034756  000005                                  RESET                   ;;CLEAR THE WORLD
034760  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
034762  000240                                  NOP                     ;;SAVE ROOM
034764  000240                                  NOP                     ;;FOR
034766  000240                                  NOP                     ;;ACT11
034770                                  $DOAGN:
034770  104400                                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
034772  042716  000020                          BIC     #20,(SP)        ;;CLEAR THE "T" BIT
034776  032777  010000  144134                  BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
```

```
4341  035004  001005                        BNE     1$              ;;BR IF NO
4342  035006  005137  035032                COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
4343  035012  100402                        BMI     1$              ;;BR IF NO
4344  035014  052716  000020                BIS     #20,(SP)        ;;SET TRACE TRAP
4345  035020  012746  035026        1$      MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
4346  035024  000002                $RTRN   RTI                     ;;RETURN--THIS IS CHANGED TO
4347                                                                ;;AN "RTT" IF "RTT" IS A LEGAL
4348                                                                ;;INSTRUCTION
4349  035026                        $LOOP:
4350  035026  000137                        JMP     @(PC)+          ;;RETURN
4351  035030  020466                $RTNAD  .WORD   LOOP
4352  035032  000000                $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
4353  035034     377      377      000    $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
4354           035040                        .EVEN
4355                                 .SBTTL  SCOPE HANDLER ROUTINE
4356
4357                                 ;;**************************************************************
4358                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS.  IT WILL INCREMENT
4359                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
4360                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4361                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4362                                 ;*SW14=1        LOOP ON TEST
4363                                 ;*SW11=1        INHIBIT ITERATIONS
4364                                 ;*SW09=1        LOOP ON ERROR
4365                                 ;*SW08=1        LOOP ON TEST IN SWR<7:0>
4366                                 ;*CALL
4367                                 ;*      SCOPE           ;;SCOPE=IOT
4368
4369  035040                        $SCOPE:
4370  035040  104410                        CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
4371  035042  032777  040000  144070 1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
4372  035050  001114                        BNE     $OVER           ;;YES IF SW14=1
4373                                 ;#####START OF CODE FOR THE XOR TESTER#####
4374  035052  000416                $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4375                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
4376  035054  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4377  035060  012737  035100  000004        MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
4378  035066  005737  177060                TST     @#177060        ;;TIME OUT ON XOR?
4379  035072  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4380  035076  000463                        BR      $SVLAD          ;;GO TO THE NEXT TEST
4381  035100  022626                5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
4382  035102  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4383  035106  000423                        BR      7$              ;;LOOP ON THE PRESENT TEST
4384  035110                        6$: ;#####END OF CODE FOR THE XOR TESTER#####
4385  035110  032777  000400  144022        BIT     #BIT08,@SWR     ;;LOOP ON SPEC.  TEST?
4386  035116  001404                        BEQ     2$              ;;BR IF NO
4387  035120  127737  144014  001102        CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
4388  035126  001466                        BEQ     $OVER           ;;BR IF YES
4389  035130  105737  001103        2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
4390  035134  001421                        BEQ     3$              ;;BR IF NO
4391  035136  123737  001115  001103        CMPB    $ERMAX,$ERFLG   ;;MAX.  ERRORS FOR THIS TEST OCCURRED?
4392  035144  101015                        BHI     3$              ;;BR IF NO
4393  035146  032777  001000  143764        BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
4394  035154  001404                        BEQ     4$              ;;BR IF NO
4395  035156  013737  001110  001106 7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
4396  035164  000446                        BR      $OVER
```

```
4397  035166  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
4398  035172  005037  001212                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4399  035176  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
4400  035200  032777  004000  143732  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
4401  035206  001011                          BNE     1$              ;;BR IF YES
4402  035210  005737  001234                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
4403  035214  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
4404  035216  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
4405  035222  023737  001212  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
4406  035230  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
4407  035232  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
4408  035240  013737  035316  001212          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
4409  035246  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
4410  035252  113737  001102  001232          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
4411  035260  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
4412  035264  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
4413  035270  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4414  035274  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4415  035302  013777  001102  143632  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
4416  035310  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
4417  035314  000002                          RTI                     ;;FIXES PS
4418  035316  000200          $MXCNT: 200                             ;;MAX  NUMBER OF ITERATIONS
4419                           $BTTL  ERROR HANDLER ROUTINE
4420
4421                          ;;**********************************************************************
4422                          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4423                          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4424                          ;*AND GO TO ERRTYP ON ERROR
4425                          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE
4426                          ;*SW15=1         HALT ON ERROR
4427                          ;*SW13=1         INHIBIT ERROR TYPEOUTS
4428                          ;*SW10=1         BELL ON ERROR
4429                          ;*SW09=1         LOOP ON ERROR
4430                          ;*CALL
4431                          ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4432
4433  035320                  $ERROR:
4434  035320  104410                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
4435  035322  010037  001162          MOV     R0,$REG0                ;SAVE THE CONTENTS OF R0
4436  035326  010137  001164          MOV     R1,$REG1                ;SAVE THE CONTENTS OF R1
4437  035332  010237  001166          MOV     R2,$REG2                ;SAVE THE CONTENTS OF R2
4438  035336  010337  001170          MOV     R3,$REG3                ;SAVE THE CONTENTS OF R3
4439  035342  010437  001172          MOV     R4,$REG4                ;SAVE THE CONTENTS OF R4
4440  035346  010537  001174          MOV     R5,$REG5                ;SAVE THE CONTENTS OF R5
4441  035352  113737  001102  001352  MOVB    $TSTNM,TESTNO           ;SAVE THE TEST NUMBER
4442  035360  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
4443  035364  001775                  BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
4444  035366  013777  001102  143546  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
4445  035374  032777  002000  143536  BIT     #BIT10,@SWR             ;;BELL ON ERROR?
4446  035402  001402                  BEQ     1$                      ;;NO - SKIP
4447  035404  104401  001216          TYPE    ,$BELL                  ;;RING BELL
4448  035410  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
4449  035414  011637  001116          MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
4450  035420  162737  000002  001116  SUB     #2,$ERRPC
4451  035426  117737  143464  001114  MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
4452  035434  032777  020000  143476  BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
```

```
4453  035442  001004                          BNE     20$             ;;SKIP TYPEOUTS
4454  035444  004737  035556                  JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
4455  035450  104401  001223                  TYPE    ,$CRLF
4456  035454                          20$
4457  035454  122737  000001  001246          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4458  035462  001007                          BNE     2$              ;;NO, SKIP APT ERROR REPORT
4459  035464  113737  001114  035476          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
4460  035472  004737  040110                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
4461  035476     000          21$     BYTE    0
4462  035477     000                  BYTE    0
4463  035500  000777          22$     BR      22$             ;;APT ERROR LOOP
4464  035502  005777  143432  2$      TST     @SWR            ;;HALT ON ERROR
4465  035506  100002                  BPL     3$              ;;SKIP IF CONTINUE
4466  035510  000000                  HALT                    ;;HALT ON ERROR!
4467  035512  104410                  CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
4468  035514  032777  001000  143416  3$      BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
4469  035522  001402                  BEQ     4$              ;;BR IF NO
4470  035524  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
4471  035530  005737  001214          4$      TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
4472  035534  001402                  BEQ     5$              ;;BR IF NONE
4473  035536  013716  001214                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
4474  035542                          5$
4475  035542  022737  034760  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
4476  035550  001001                  BNE     6$              ;;BRANCH IF NO
4477  035552  000000                  HALT                    ;;YES
4478  035554                          6$
4479  035554  000002                  RTI                     ;;RETURN
4480                                   SBTTL   ERROR MESSAGE TYPEOUT ROUTINE
4481
4482                                  ;;*********************************************************************
4483                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4484                                  ;*ERROR IS TO BE REPORTED.  IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4485                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR
4486                                  ;*
4487                                  ;*NOTES:
4488                                  ;*1)    THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
4489                                  ;*      FOR "EM", "DM", AND "DT".
4490                                  ;*2)    TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
4491                                  ;*3)    FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4492                                  ;*4)    THE AVAILABLE FORMATS FOR TYPING DATA ARE
4493                                  ;*      DF      FORMAT
4494                                  ;*      0       TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
4495                                  ;*      1       TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
4496                                  ;*      2       TYPE A 16 DIGIT BINARY NUMBER
4497                                  ;*      3       TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
4498                                  ;*
4499
4500  035556                          ERRTYP
4501  035556  104401  001223                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
4502  035562  010046                          MOV     R0,-(SP)        ;SAVE R0
4503  035564  005000                          CLR     R0              ;PICKUP THE ITEM INDEX
4504  035566  153700  001114                  BISB    @#$ITEMB,R0
4505  035572  001004                          BNE     1$              ;IF ITEM NUMBER IS ZERO  JUST
4506                                                                  ;TYPE THE PC OF THE ERROR
4507  035574  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
4508                                                                  ;;ERROR ADDRESS
```

```
4509  035600  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4510  035602  000471                  BR      13$             ;GET OUT
4511  035604  005300          1$      DEC     R0              ;ADJUST THE INDEX SO THAT IT WILL
4512  035606  006300                  ASL     R0              ;       WORK FOR THE ERROR TABLE
4513  035610  006300                  ASL     R0
4514  035612  006300                  ASL     R0
4515  035614  062700  001412          ADD     #$ERRTB,R0      ;FORM TABLE POINTER
4516  035620  012037  035630          MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
4517  035624  001404                  BEQ     3$              ;SKIP TYPEOUT IF NO POINTER
4518  035626  104401                  TYPE                    ;TYPE THE "ERROR MESSAGE"
4519  035630  000000          2$       WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
4520  035632  104401  001223          TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
4521  035636  012037  035646  3$      MOV     (R0)+,4$        ;PICKUP "DATA HEADER" POINTER
4522  035642  001404                  BEQ     5$              ;SKIP TYPEOUT IF 0
4523  035644  104401                  TYPE                    ;TYPE THE "DATA HEADER"
4524  035646  000000          4$       WORD   0               ;"DATA HEADER" POINTER GOES HERE
4525  035650  104401  001223          TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
4526  035654  010146          5$      MOV     R1,-(SP)        ;SAVE R1
4527  035656  012001                  MOV     (R0)+,R1        ;PICKUP "DATA TABLE" POINTER
4528  035660  001441                  BEQ     12$             ;BR IF NO DATA TO BE TYPED
4529  035662  012000                  MOV     (R0)+,R0        ;PICKUP "DATA FORMAT" POINTER
4530  035664  105710          6$      TSTB    (R0)            ;IS IT FORMAT 0?
4531  035666  001003                  BNE     7$              ;BR IF NO
4532
4533                          ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4534  035670  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4535  035672  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4536  035674  000425                  BR      11$
4537
4538                          ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4539  035676  121027  000001  7$      CMPB    (R0),#1         ;IS IT FORMAT 1?
4540  035702  001003                  BNE     8$              ;BRANCH IF NO
4541  035704  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4542  035706  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
4543  035710  000417                  BR      11$
4544
4545                          ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
4546  035712  121027  000002  8$      CMPB    (R0),#2         ;IS IT FORMAT 2?
4547  035716  001003                  BNE     9$              ;BRANCH IF NO
4548  035720  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4549  035722  104406                  TYPBN                   ;;GO TYPE--BINARY ASCII
4550  035724  000411                  BR      11$
4551
4552                          ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
4553  035726  012146          9$      MOV     (R1)+,-(SP)     ;PUT ADDRESS OF FIRST LOC  ON STACK
4554  035730  004737  041162          JSR     PC,$DB2O        ;CONVERT TWO LOCS  TO AN ASCII STRING
4555  035734  062716  000005          ADD     #5,(SP)         ;ONLY NEED 6 CHARACTERS NOT 11
4556  035740  012637  035746          MOV     (SP)+,10$       ;PUT ADDRESS OF ASCII CHARS  AT 10$
4557  035744  104401                  TYPE                    ;TYPE OCTAL VALUE OF 18-BIT BINARY NO
4558  035746  000000          10$      WORD   0
4559
4560  035750  005711          11$     TST     (R1)            ;IS THERE ANOTHER NUMBER?
4561  035752  001404                  BEQ     12$             ;BR IF NO
4562  035754  104401  035776          TYPE    ,14$            ;TYPE TWO(2) SPACES
4563  035760  105720                  TSTB    (R0)+           ;POINT TO NEW "DATA FORMAT"
4564  035762  000740                  BR      6$              ;LOOP
```

```
4565
4566  035764  012601                  12$     MOV     (SP)+,R1        ;RESTORE R1
4567  035766  012600                  13$     MOV     (SP)+,R0        ;RESTORE R0
4568  035770  104401  001223                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
4569  035774  000207                          RTS     PC              ;RETURN
4570  035776  020040      000         14$      ASCIZ  /  /            ;TWO(2) SPACES
4571          036002                           EVEN
4572
```

```
                                 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 °SW
                                ;*      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
                                ;*
                                ;;*****************************************************************
036002  033727  177776  000020  TOFF    BIT     PSW,#TBIT       ;IS THE T-BIT SET IN THE PSW?
036010  001411                          BEQ     1$              ;EXIT IF NO
036012  013746  177776                  MOV     PSW,-(SP)       ;PUSH PRESENT PSW ON THE STACK
036016  011637  001372                  MOV     (SP),TBITPS     ;ALSO SAVE IT IN "TBITPS" FOR
                                                                ;RESTORING LATER
036022  042716  000020                  BIC     #TBIT,(SP)      ;CLEAR THE T-BIT (BIT 4) IN THE PSW
036026  012746  036034                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
036032  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT OFF
036034  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
                                ;*
                                ;;*****************************************************************
036036  033727  001372  000020  TON     BIT     TBITPS,#TBIT    ;WAS T-BIT ON IN THE PREVIOUS PSW?
036044  001410                          BEQ     1$              ;EXIT IF NO
036046  013746  001372                  MOV     TBITPS,-(SP)    ;PUSH PREVIOUS PSW ON THE STACK
036052  012737  000340  001372          MOV     #340,TBITPS     ;RESET THE "TBITPS" LOCATION
036060  012746  036066                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
036064  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT RESTORED
036066  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.
                                ;*
                                ;;*****************************************************************
036070  012702  000010  SETREG  MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
036074  012701  172300          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
036100  012721  177777  1$      MOV     #-1,(R1)+       ;SET BITS IN KERNEL PDR TO 1
036104  077203                  SOB     R2,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
036106  012702  000010          MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
036112  012701  172340          MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
036116  012721  177777  2$      MOV     #-1,(R1)+       ;SET BITS IN A KERNEL PAR TO 1
036122  077203                  SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PAR S LOADED
```

```
 4629  036124  012702  000010                MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
 4630  036130  012701  177600                MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
 4631  036134  012721  177777      3$        MOV     #-1,(R1)+       ;SET BITS IN A USER PDR TO 1
 4632  036140  077203                        SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
 4633  036142  012702  000010                MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
 4634  036146  012701  177640                MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
 4635  036152  012721  177777      4$        MOV     #-1,(R1)+       ;SET BITS IN A USER PAR TO 1
 4636  036156  077203                        SOB     R2,4$           ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
 4637  036160  000207                        RTS     PC              ;RETURN TO TEST
 4638
 4639                                  .SBTTL  READ & COMPARE KERNEL & USER PAR/PDR'S
 4640                                  ;;**********************************************************************
 4641                                  ;*
 4642                                  ;*      THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
 4643                                  ;*      READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
 4644                                  ;*      WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS
 4645                                  ;*      ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
 4646                                  ;*      BE REPORTED BY THIS SUBROUTINE.
 4647                                  ;*
 4648                                  ;;**********************************************************************
 4649  036162                          CMPREG
 4650  036162  012701  172300                MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
 4651  036166  012704  000010                MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4652  036172  012705  077416                MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
 4653  036176  011102      1$        MOV     (R1),R2         ;READ A KERNEL PDR INTO R2
 4654  036200  020205                        CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AL EXPECTED?
 4655  036202  001403                        BEQ     2$              ;BRANCH IF YES
 4656  036204  020100                        CMP     R1,R0           ;WAS IT THE REG.  THAT WAS CLEARED?
 4657  036206  001401                        BEQ     2$              ;BRANCH IF YES
 4658  036210  104016                        ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
 4659                                                                ;FOR TIGHTER SCOPE LOOP
 4660                                                                ;REPLACE ERROR CALL WITH
 4661                                                                ;AN "RTS PC" = 000207
 4662  036212  062701  000002      2$        ADD     #2,R1           ;FORM NEXT KERNEL PDR ADDRESS
 4663  036216  077411                        SOB     R4,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
 4664  036220  012701  172340                MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
 4665  036224  012704  000010                MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4666  036230  012705  007777                MOV     #7777,R5        ;PUT EXPECTED PAR CONTENTS IN R5
 4667  036234  011102      3$        MOV     (R1),R2         ;READ A KERNEL PAR INTO R2
 4668  036236  020205                        CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
 4669  036240  001403                        BEQ     4$              ;BRANCH IF YES
 4670  036242  020100                        CMP     R1,R0           ;WAS IT THE REG  THAT WAS CLEARED?
 4671  036244  001401                        BEQ     4$              ;BRANCH IF YES
 4672  036246  104016                        ERROR   16              ;A PAR WAS EFFECTED BY CLEARINg A D FFENENT PAR/PD?
 4673                                                                ;FOR TIGHTER SCOPE LOOP
 4674                                                                ;REPLACE ERROR CALL WITH
 4675                                                                ;AN "RTS PC" = 000207
 4676  036250  062701  000002      4$        ADD     #2,R1           ;FORM NEXT KERNEL PAR ADDRESS
 4677  036254  077411                        SOB     R4,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
 4678  036256  012701  177600                MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST USER PDR IN R1
 4679  036262  012704  000010                MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
 4680  036266  012705  077416                MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
 4681  036272  011102      5$        MOV     (R1),R2         ;READ A USER PDR INTO R2
 4682  036274  020205                        CMP     R2,R5           ;ARE ALL WRITABLE BITS SET AS EXPECTED?
 4683  036276  001403                        BEQ     6$              ;BRANCH IF YES
 4684  036300  020100                        CMP     R1,R0           ;WAS IT THE REG  THAT WAS CLEARED?
```

```
4685  036302  001401                          BEQ     6$              ;BRANCH IF YES
4686  036304  104016                          ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4687                                                                  ;FOR TIGHTER SCOPE LOOP
4688                                                                  ;REPLACE ERROR CALL WITH
4689                                                                  ;AN "RTS PC" = 000207
4690  036306  062701  000002          6$      ADD     #2,R1           ;FORM NEXT USER PDR ADDRESS
4691  036312  077411                          SOB     R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4692  036314  012701  177640                  MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
4693  036320  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4694  036324  012705  007777                  MOV     #7777,R5        ;PUT EXPECTED PAR CONTENTS IN R5
4695  036330  011102                  7$      MOV     (R1),R2         ;READ A USER PAR INTO R2
4696  036332  020205                          CMP     R2,R5           ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4697  036334  001403                          BEQ     8$              ;BRANCH IF YES
4698  036336  020100                          CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4699  036340  001401                          BEQ     8$              ;BRANCH IF YES
4700  036342  104016                          ERROR   16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4701                                                                  ;FOR TIGHTER SCOPE LOOP
4702                                                                  ;REPLACE ERROR CALL WITH
4703                                                                  ;AN "RTS PC" = 000207
4704  036344  062701  000002          8$      ADD     #2,R1           ;FORM NEXT USER PAR ADDRESS
4705  036350  077411                          SOB     R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4706  036352  000207                          RTS     PC              ;RETURN TO TEST
4707
4708                                  .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
4709                                  ;;**********************************************************************
4710                                  ;*
4711                                  ;*      THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
4712                                  ;*      (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
4713                                  ;*      PAGE ADDRESS REGISTER (PAR).  THE SAME METHOD USED BY THE MEMORY
4714                                  ;*      MANAGEMENT LOGIC IS USED.  VBA <15:13> SELECTS WHICH PAR/PDR
4715                                  ;*      IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
4716                                  ;*      TO PAR <11:00> TO GIVE PBA <17:6>   BITS <17:16> OF THE
4717                                  ;*      PHYSICAL ADDRESS ARE LEFT IN LOC  "PBAHI" AND BITS <15 00>
4718                                  ;*      ARE LEFT IN LOC. "PBALO"   THE PSW'S "CURRENT MODE" BITS
4719                                  ;*      ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S   THE ROUTINE
4720                                  ;*      IS ENTERED WITH LOC  "VIRT1" CONTAINING THE 16-BIT VIRTUAL
4721                                  ;*      ADDRESS
4722                                  ;*
4723                                  ;;**********************************************************************
4724
4725  036354  012702  172340          FORMPA  MOV     #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4726  036360  032737  140000  177776          BIT     #140000,PSW     ;IN USER MODE?
4727  036366  001402                          BEQ     1$              ;BRANCH IF NO
4728  036370  012702  177640                  MOV     #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
4729  036374  013700  001402          1$      MOV     VIRT1,R0        ;LOAD VIRTUAL ADDR  (VBA) INTO R0
4730  036400  072027  177764                  ASH     #-14,R0         ;GET BITS <15 13> DOWN TO BITS <3 1>
4731  036404  042700  177761                  BIC     #177761,R0      ;MASK OF ALL BITS BUT BITS <3 1>
4732  036410  060002                          ADD     R0,R2           ;ADD OFFSET TO BASE PAR ADDRESS
4733  036412  011200                          MOV     (R2),R0         ;GET BITS <11 00> FROM APPROPRIATE PAR
4734  036414  010002                          MOV     R0,R2           ;COPY PAR BITS <11 00> INTO R2
4735  036416  013737  001402  001406          MOV     VIRT1,PBALO     ;PUT VIRTUAL ADDR  IN LOC  "PBALO"
4736  036424  042737  160000  001406          BIC     #160000,PBALO   ;CLEAR OFF BITS <15 13> OF ORIGINAL VBA
4737  036432  072227  177766                  ASH     #-12,R2         ;GET PAR <11 00> DOWN TO BITS <1 0> OF R2
4738  036436  042702  177774                  BIC     #177774,R2      ;CLEAR OFF ALL BITS BUT BITS <1 0>
4739  036442  072027  000006                  ASH     #6,R0           ;SHIFT PAR<9 0> TO <15 6> OF R0
4740  036446  042700  000077                  BIC     #77,R0          ;CLEAR BITS <5 0> OF R0
```

```
 4741  036452  060037  001406                ADD     R0,PBALO        ;IN EFFECT, ADD VBA(12 0) TO PAR(9 0)
 4742                                                                  ;(PAR(9 0) IN BITS (15 6) OF R0)
 4743  036456  005502                        ADC     R2              ;ADD ANY CARRY TO R2
 4744  036460  010237  001410                MOV     R2,PBAHI        ;PUT BITS (17 16) OF PHYSICAL ADDR  IN PBAHI
 4745  036464  000207                        RTS     PC              ;RETURN TO PROGRAM
 4746
```

```
                                        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
036466  022737  000176  001140  $CKSWR  CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
036474  001114                          BNE     15$             ;;BRANCH IF NO
036476  105777  142442                  TSTB    @$TKS           ;;CHAR THERE?
036502  100111                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
036504  117746  142436                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
036510  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
036514  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
036520  001102                          BNE     15$             ;;NO, RETURN TO USER
036522  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
036530  001476                          BEQ     15$             ;;BRANCH IF YES

036532  104401  037430                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G ( G)
036536  104401  037435  $GTSWR  TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
036542  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
036546  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
036550  104401  037446                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
036554  005046          19$     CLR     -(SP)           ;;CLEAR COUNTER
036556  005046                          CLR     -(SP)           ;;THE NEW SWR
036560  105777  142360          7$      TSTB    @$TKS           ;;CHAR THERE?
036564  100375                          BPL     7$              ;;IF NOT TRY AGAIN

036566  117746  142354                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
036572  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

036576  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
036602  001015                          BNE     9$              ;;BRANCH IF NOT
036604  104401  037416                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C ( C)
036610  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
036614  123727  001135  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
036622  001003                          BNE     8$              ;;BRANCH IF NO
036624  012777  000100  142312          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
036632  000137  037460          8$      JMP     CNTRLC          ;;CONTROL-C RESTART

036636  021627  000025          9$      CMP     (SP),#25        ;;IS IT A CONTROL-U?
036642  001005                          BNE     10$             ;;BRANCH IF NOT
036644  104401  037423                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U ( U)
036650  062706  000006          20$     ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
036654  000737                          BR      19$             ;;LET'S TRY IT AGAIN

036656  021627  000015          10$     CMP     (SP),#15        ;;IS IT A <CR>?
036662  001012                          BNE     16$             ;;BRANCH IF NO
036664  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
036670  001403                          BEQ     11$             ;;BRANCH IF YES
036672  016677  000002  142240          MOV     2(SP),@SWR      ;;SAVE NEW SWR
```

```
4803  036700  062706  000006          11$     ADD     #6,SP           ;;CLEAR UP STACK
4804  036704  104401  001223          14$     TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
4805  036710  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
4806  036716  001003                          BNE     15$             ;;BRANCH IF NOT
4807  036720  012777  000100  142216          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
4808  036726  000002                  15$     RTI                     ;;RETURN
4809  036730  004737  040022          16$     JSR     PC,$TYPEC       ;;ECHO CHAR
4810  036734  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
4811  036740  002420                          BLT     18$             ;;BRANCH IF YES
4812  036742  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
4813  036746  003015                          BGT     18$             ;;BRANCH IF YES
4814  036750  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
4815  036754  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
4816  036760  001403                          BEQ     17$             ;;BRANCH IF YES
4817  036762  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
4818  036764  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
4819  036766  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE
4820  036770  005266  000002          17$     INC     2(SP)           ;;KEEP COUNT OF CHAR
4821  036774  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
4822  037000  000667                          BR      7$              ;;GET THE NEXT ONE
4823  037002  104401  001222          18$     TYPE    ,$QUES          ;;TYPE ?<CR><LF>
4824  037006  000720                          BR      20$             ;;SIMULATE CONTROL-U
4825                                  DSABL   LSB
4826
4827                                  ;;**********************************************************************
4828                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4829                                  ;*CALL
4830                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4831                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4832                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
4833                                  ;
4834
4835
4836  037010  011646                  $RDCHR  MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4837  037012  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
4838  037020  105777  142120          1$      TSTB    @$TKS           ;;WAIT FOR
4839  037024  100375                          BPL     1$              ;;A CHARACTER
4840  037026  117766  142114  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
4841  037034  042766  177600  000004          BIC     # C(177),4(SP)  ;;GET RID OF JUNK IF ANY
4842  037042  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
4843  037050  001013                          BNE     3$              ;;BRANCH IF NO
4844  037052  105777  142066          2$      TSTB    @$TKS           ;;WAIT FOR A CHARACTER
4845  037056  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
4846  037060  117746  142062                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
4847  037064  042716  177600                  BIC     # C177,(SP)     ;;MAKE IT 7-BIT ASCII
4848  037070  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
4849  037074  001366                          BNE     2$              ;;IF NOT DISCARD IT
4850  037076  000750                          BR      1$              ;;YES, RESUME
4851  037100  026627  000004  000140  3$      CMP     4(SP),#140      ;;IS IT UPPER CASE?
4852  037106  002407                          BLT     4$              ;;BRANCH IF YES
4853  037110  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
4854  037116  003003                          BGT     4$              ;;BRANCH IF YES
4855  037120  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
4856  037126  000002                  4$      RTI                     ;;GO BACK TO USER
4857                                  ;;**********************************************************************
4858                                  ;*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

037130  010346                  $RDLIN  MOV     R3,-(SP)        ;;SAVE R3
037132  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
037134  012703  037406          1$      MOV     #$TTYIN,R3      ;;GET ADDRESS
037140  022703  037416          2$      CMP     #$TTYIN+8 ,R3   ;;BUFFER FULL?
037144  101467                          BLOS    4$              ;;BR IF YES
037146  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
037150  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
037152  122713  000003                  CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
037156  001006                          BNE     10$             ;;BRANCH IF NO
037160  104401  037416                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C ( C)
037164  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
037166  012603                          MOV     (SP)+,R3        ;;RESTORE R3
037170  000137  037460                  JMP     CNTRLC          ;;GOTO CONTROL-C RESTART
037174  122713  000177          10$     CMPB    #177,(R3)       ;;IS IT A RUBOUT
037200  001022                          BNE     5$              ;;BR IF NO
037202  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
037204  001007                          BNE     6$              ;;BR IF NO
037206  112737  000134  037404          MOVB    #'\,9$          ;;TYPE A BACK SLASH
037214  104401  037404                  TYPE    ,9$
037220  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
037224  006303                  6$      DEC     R3              ;;BACKUP BY ONE
037226  020327  037406                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
037232  103434                          BLO     4$              ;;BR IF YES
037234  111337  037404                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR
037240  104401  037404                  TYPE    ,9$             ;;GO TYPE
037244  000735                          BR      2$              ;;GO READ ANOTHER CHAR
037246  005716                  5$      TST     (SP)            ;;RUBOUT KEY SET?
037250  001406                          BEQ     7$              ;;BR IF NO
037252  112737  000134  037404          MOVB    #'\,9$          ;;TYPE A BACK SLASH
037260  104401  037404                  TYPE    ,9$
037264  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
037266  122713  000025          7$      CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
037272  001003                          BNE     8$              ;;BR IF NO
037274  104401  037423                  TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
037300  000715                          BR      1$              ;;GO START OVER
037302  122713  000022          8$      CMPB    #22,(R3)        ;;IS CHARACTER A " R"?
037306  001011                          BNE     3$              ;;BRANCH IF NO
037310  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
037312  104401  001223                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
037316  104401  037406                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
037322  000706                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
037324  104401  001222          4$      TYPE    ,$QUES          ;;TYPE A '?'
037330  000701                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
037332  111337  037404          3$      MOVB    (R3),9$         ;;ECHO THE CHARACTER
037336  104401  037404                  TYPE    ,9$
037342  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
037346  001274                          BNE     2$              ;;LOOP IF NOT RETURN
037350  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
037354  104401  001224                  TYPE    ,$LF            ;;TYPE A LINE FEED
037360  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
037362  012603                          MOV     (SP)+,R3        ;;RESTORE R3
```

```
 4915  037364  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
 4916  037366  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
 4917  037374  012766  037406  000004          MOV     #STTYIN,4(SP)
 4918  037402  000002                          RTI                     ;;RETURN
 4919  037404     000                  9$      .BYTE   0               ;;STORAGE FOR ASCII CHAR  TO TYPE
 4920  037405     000                          .BYTE   0               ;;TERMINATOR
 4921  037406  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
 4922  037416  041536  005015     000 $CNTLC: .ASCIZ  / C/<15><12>    ;;CONTROL "C"
 4923  037423     136  006525  000012 $CNTLU: .ASCIZ  / U/<15><12>    ;;CONTROL "U"
 4924  037430  043536  005015     000 $CNTLG: .ASCIZ  / G/<15><12>    ;;CONTROL "G"
 4925  037435     015  051412  051127 $MSWR:  .ASCIZ  <15><12>/SWR = /
 4926  037442  036440  000040
 4927  037446  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW = /
 4928  037454  020075     000
 4929          037460                  .EVEN
 4930
 4931                                  .SBTTL  CONTROL-C SERVICING ROUTINE
 4932
 4933                                  ;*      THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
 4934                                  ;*      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
                                        ;*
037460  013737  001234  001210  CNTRLC: MOV     $PASS,$TMP5     ;GET THE VALUE OF "$PASS"
037466  005237  001210                  INC     $TMP5           ;FORM CURRENT PASS NO.
037472  104401  037537                  TYPE    ,CMSG           ;TYPE THE TEST STOPS MESSAGE
037476  113737  001102  037532          MOVB    $TSTNM,1$       ;SAVE THE TEST NUMBER
037504  013746  037532                  MOV     1$,-(SP)        ;;SAVE 1$ FOR TYPEOUT
037510  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
037512  104401  037534                  TYPE    ,2$             ;TYPE 2 SPACES
037516  013746  001210                  MOV     $TMP5,-(SP)     ;;SAVE $TMP5 FOR TYPEOUT
037522  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
037524  104407                          GTSWR                   ;ASK FOR NEW SWR VALUE
037526  000137  034554                  JMP     $EOP+2          ;CONTINUE AT $EOP+2
037532  000000                  1$:     .WORD   0               ;BUFFER FOR TEST NUMBER
037534  020040      000         2$:     .ASCIZ  /  /            ;TWO SPACES AND THE STOP MESSAGE
037537     112  046525  044520  CMSG:   .ASCII  /JUMPING TO END-OF-PASS/<15><12>
037544  043516  052040  020117
037562  047105  026504  043117
037560  060055  051501  006523
037566     012
037567     124  051505  047124          .ASCIZ  /TESTNO  PASSNO/<15><12>
037574  020117  050040  051501
037602  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
                                ;*
037610  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
037614  100002                          BPL     1$              ;;BR IF YES
037616  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
037620  000430                          BR      3$              ;;LEAVE
037622  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
037624  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
037630  122737  000001  001246          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
037636  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
037640  132737  000100  001247          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
037646  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
037650  010037  037660                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
037654  004737  040100                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
```

```
4991  037660  000000                    61$     WORD    0               ;;MESSAGE ADDRESS
4992  037662  132737  000040  001247    62$     BITB    #APTCSUP $ENVM  ;;APT CONSOLE SUPPRESSED
4993  037670  001003                            BNE     60$             ;;YES, SKIP TYPE OUT
4994  037672  112046                    2$      MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4995  037674  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
4996  037676  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
4997  037700  012600                    60$     MOV     (SP)+,R0        ;;RESTORE R0
4998  037702  062716  000002            3$      ADD     #2,(SP)         ;;ADJUST RETURN PC
4999  037706  000002                            RTI                     ;;RETURN
5000  037710  122716  000011            4$      CMPB    #HT,(SP)        ;;BRANCH IF <HT>
5001  037714  001430                            BEQ     8$
5002  037716  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
5003  037722  001006                            BNE     5$
5004  037724  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
5005  037726  104401                            TYPE                    ; TYPE A CR AND LF
5006  037730  001223                            $CRLF
5007  037732  105037  040066                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
5008  037736  000755                            BR      2$              ;;GET NEXT CHARACTER
5009  037740  004737  040022            5$      JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
5010  037744  123726  001156            6$      CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS ?
5011  037750  001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
5012  037752  013746  001154                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS  NEEDED
5013                                                                    ;;AND THE NULL CHAR
5014  037756  105366  000001            7$      DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
5015  037762  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
5016  037764  004737  040022                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
5017  037770  105337  040066                    DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
5018  037774  000770                            BR      7$              ;;LOOP
5019
5020                                    ;HORIZONTAL TAB PROCESSOR
5021
5022  037776  112716  000040            8$      MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
5023  040002  004737  040022            9$      JSR     PC,$TYPEC       ;;TYPE A SPACE
5024  040006  132737  000007  040066            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
5025  040014  001372                            BNE     9$              ;;TAB STOP
5026  040016  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
5027  040020  000724                            BR      2$              ;;GET NEXT CHARACTER
5028  040022  105777  141122            $TYPEC  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
5029  040026  100375                            BPL     $TYPEC
5030  040030  116677  000002  141114            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG
5031  040036  122766  000015  000002            CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
5032  040044  001003                            BNE     1$              ;;BRANCH IF NO
5033  040046  105037  040066                    CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
5034  040052  000406                            BR      $TYPEX          ;;EXIT
5035  040054  122766  000012  000002    1$      CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
5036  040062  001402                            BEQ     $TYPEX          ;;BRANCH IF YES
5037  040064  105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
5038  040066  000000            $CHARCNT: WORD  0               ;;CHARACTER COUNT STORAGE
5039  040070  000207            $TYPEX. RTS     PC
5040
5041                                     SBTTL  APT COMMUNICATIONS ROUTINE
5042
5043                                    ;;**********************************************************************
5044  040072  112737  000001  040336    $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
5045  040100  112737  000001  040334    $ATY3   MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
5046  040106  000403                            BR      $ATYC
```

```
5047  040110  112737  000001  040336  $ATY4   MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
5048  040116                          $ATYC
5049  040116  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5050  040120  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5051  040122  105737  040334                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
5052  040126  001450                          BEQ     5$              ;;IF NOT  BR
5053  040130  122737  000001  001246          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
5054  040136  001031                          BNE     3$              ;;IF NOT  BR
5055  040140  132737  000100  001247          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
5056  040146  001425                          BEQ     3$              ;;IF NOT  BR
5057  040150  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR
5058  040154  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR
5059  040162  005737  001226          1$      TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
5060  040166  001375                          BNE     1$              ;;IF NOT  WAIT
5061  040170  010037  001242                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
5062  040174  105720                  2$      TSTB    (R0)+           ;;FIND END OF MESSAGE
5063  040176  001376                          BNE     2$
5064  040200  163700  001242                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
5065  040204  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
5066  040206  010037  001244                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
5067  040212  012737  000004  001226          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
5068  040220  000413                          BR      5$
5069  040222  017637  000004  040246  3$      MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
5070  040230  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
5071  040236  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
5072  040242  004737  037610                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
5073  040246  000000                  4$       WORD    0
5074  040250                          5$
5075  040250  105737  040336          10$     TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
5076  040254  001416                          BEQ     12$             ;;IF NOT  BR
5077  040256  005737  001246                  TST     $ENV            ;;RUNNING UNDER APT?
5078  040262  001413                          BEQ     12$             ;;IF NOT:  BR
5079  040264  005737  001226          11$     TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
5080  040270  001375                          BNE     11$             ;;IF NOT.  WAIT
5081  040272  017637  000004  001230          MOV     @4(SP),$FATAL   ;;GET ERROR #
5082  040300  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR
5083  040306  005237  001226                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
5084  040312  105037  040336          12$     CLRB    $FFLG           ;;CLEAR FATAL FLAG
5085  040316  105037  040335                  CLRB    $LFLG           ;;CLEAR LOG FLAG
5086  040322  105037  040334                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
5087  040326  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5088  040330  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
5089  040332  000207                          RTS     PC              ;;RETURN
5090  040334     000          $MFLG   .BYTE   0               ;;MESSG  FLAG
5091  040335     000          $LFLG:   BYTE   0               ;;LOG FLAG
5092  040336     000          $FFLG:   BYTE   0               ;;FATAL FLAG
5093          040340                  .EVEN
5094          000200          APTSIZE=200
5095          000001          APTENV=001
5096          000100          APTSPOOL=100
5097          000040          APTCSUP=040
5098                           .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
5099
5100                           ;;**************************************************************
5101                           ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
5102                           ;*BINARY-ASCII NUMBER AND TYPE IT
```

```
                                        ,*CALL
                                        ,*      MOV     NUMBER,-(SP)    ,,NUMBER TO BE TYPED
                                        ,*      TYPBN                   ,,TYPE IT
                                        ,*
040340  010146                  $TYPBN  MOV     R1,-(SP)        ,,SAVE R1 ON THE STACK
040342  016601  000006                  MOV     6(SP),R1        ,,GET THE INPUT NUMBER
040346  000261                          SEC                     ,,SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
040350  112737  000060  040412  1$      MOVB    #'0,$BIN        ,,SET CHARACTER TO AN ASCII "0"
040356  006101                          ROL     R1              ,,GET THIS BIT
040360  001406                          BEQ     2$              ,,DONE?
040362  105537  040412                  ADCB    $BIN            ,,NO--SET THE CHARACTER EQUAL TO THIS BIT
040366  104401  040412                  TYPE    ,$BIN           ,,GO TYPE THIS BIT
040372  000241                          CLC                     ,,CLEAR "C" SO CAN KEEP TRACK OF BITS
040374  000765                          BR      1$              ,,GO DO THE NEXT BIT
040376  012601                  2$      MOV     (SP)+,R1        ,,POP THE STACK INTO R1
040400  016666  000002  000004          MOV     2(SP),4(SP)     ,;ADJUST THE STACK
040406  012616                          MOV     (SP)+,(SP)
040410  000002                          RTI                     ,,RETURN TO USER
040412    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

040414  017646  000000          $TYPOS  MOV     @(SP),-(SP)     ,;PICKUP THE MODE
040420  116637  000001  040637          MOVB    1(SP),$OFILL    ,;LOAD ZERO FILL SWITCH
040426  112637  040641                  MOVB    (SP)+,$OMODE+1  ,,NUMBER OF DIGITS TO TYPE
040432  062716  000002                  ADD     #2,(SP)         ,,ADJUST RETURN ADDRESS
040436  000406                          BR      $TYPON
040440  112737  000001  040637  $TYPOC· MOVB    #1,$OFILL       ,,SET THE ZERO FILL SWITCH
040446  112737  000006  040641          MOVB    #6,$OMODE+1     ,,SET FOR SIX(6) DIGITS
040454  112737  000005  040636  $TYPON  MOVB    #5,$OCNT        ,;SET THE ITERATION COUNT
040462  010346                          MOV     R3,-(SP)        ,,SAVE R3
040464  010446                          MOV     R4,-(SP)        ,,SAVE R4
040466  010546                          MOV     R5,-(SP)        ,,SAVE R5
040470  113704  040641                  MOVB    $OMODE+1,R4     ,,GET THE NUMBER OF DIGITS TO TYPE
```

```
5159  040474  005404                          NEG     R4
5160  040476  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX  ALLOWED
5161  040502  110437  040640                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
5162  040506  113704  040637                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
5163  040512  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
5164  040516  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
5165  040520  006105                  1$      ROL     R5              ;;ROTATE MSB INTO "C"
5166  040522  000404                          BR      3$              ;;GO DO MSB
5167  040524  006105                  2$      ROL     R5              ;;FORM THIS DIGIT
5168  040526  006105                          ROL     R5
5169  040530  006105                          ROL     R5
5170  040532  010503                          MOV     R5,R3
5171  040534  006103                  3$      ROL     R3              ;;GET LSB OF THIS DIGIT
5172  040536  105337  040640                  DECB    $OMODE          ;;TYPE THIS DIGIT?
5173  040542  100016                          BPL     7$              ;;BR IF NO
5174  040544  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
5175  040550  001002                          BNE     4$              ;;TEST FOR 0
5176  040552  005704                          TST     R4              ;;SUPPRESS THIS 0?
5177  040554  001403                          BEQ     5$              ;;BR IF YES
5178  040556  005204                  4$      INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
5179  040560  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
5180  040564  052703  000040          5$      BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
5181  040570  110337  040634                  MOVB    R3,8$           ;;SAVE FOR TYPING
5182  040574  104401  040634                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
5183  040600  105337  040636          7$      DECB    $OCNT           ;;COUNT BY 1
5184  040604  003347                          BGT     2$              ;;BR IF MORE TO DO
5185  040606  002402                          BLT     6$              ;;BR IF DONE
5186  040610  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
5187  040612  000744                          BR      2$              ;;GO DO THE LAST DIGIT
5188  040614  012605                  6$      MOV     (SP)+,R5        ;;RESTORE R5
5189  040616  012604                          MOV     (SP)+,R4        ;;RESTORE R4
5190  040620  012603                          MOV     (SP)+,R3        ;;RESTORE R3
5191  040622  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
5192  040630  012616                          MOV     (SP)+,(SP)
5193  040632  000002                          RTI                     ;;RETURN
5194  040634     000                  8$       BYTE   0               ;;STORAGE FOR ASCII DIGIT
5195  040635     000                           BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
5196  040636     000                  $OCNT    BYTE   0               ;;OCTAL DIGIT COUNTER
5197  040637     000                  $OFILL   BYTE   0               ;;ZERO FILL SWITCH
5198  040640  000000                  $OMODE.  WORD   0               ;;NUMBER OF DIGITS TO TYPE
5199                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5200
5201                                  ;;**************************************************************
5202                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5203                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT  DEPENDING ON WHETHER THE
5204                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5205                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER  LEADING ZEROS WILL ALWAYS BE
5206                                  ;*REPLACED WITH SPACES
5207                                  ;*CALL
5208                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
5209                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
5210
5211  040642                          $TYPDS.
5212  040642  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5213  040644  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5214  040646  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

```
5215  040650  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5216  040652  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5217  040654  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
5218  040660  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
5219  040664  100004                          BPL     1$              ;;BR IF INPUT IS POS.
5220  040666  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS
5221  040670  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG
5222  040676  005000                  1$      CLR     R0              ;;ZERO THE CONSTANTS INDEX
5223  040700  012703  041056                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
5224  040704  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
5225  040710  005002                  2$      CLR     R2              ;;CLEAR THE BCD NUMBER
5226  040712  016001  041046                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
5227  040716  160105                  3$      SUB     R1,R5           ;;FORM THIS BCD DIGIT
5228  040720  002402                          BLT     4$              ;;BR IF DONE
5229  040722  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
5230  040724  000774                          BR      3$
5231  040726  060105                  4$      ADD     R1,R5           ;;ADD BACK THE CONSTANT
5232  040730  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
5233  040732  001002                          BNE     5$              ;;FALL THROUGH IF 0
5234  040734  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
5235  040736  100407                          BMI     7$              ;;BR IF YES
5236  040740  106316                  5$      ASLB    (SP)            ;;MSD?
5237  040742  103003                          BCC     6$              ;;BR IF NO
5238  040744  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES-- SET THE SIGN
5239  040752  052702  000060          6$      BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
5240  040756  052702  000040          7$      BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5241  040762  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5242  040764  005720                          TST     (R0)+           ;;JUST INCREMENTING
5243  040766  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
5244  040772  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
5245  040774  003002                          BGT     8$              ;;GO TO EXIT
5246  040776  010502                          MOV     R5,R2           ;;GET THE LSD
5247  041000  000764                          BR      6$              ;;GO CHANGE TO ASCII
5248  041002  105726                  8$      TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
5249  041004  100003                          BPL     9$              ;;BR IF NO
5250  041006  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
5251  041014  105013                  9$      CLRB    (R3)            ;;SET THE TERMINATOR
5252  041016  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
5253  041020  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
5254  041022  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5255  041024  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5256  041026  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
5257  041030  104401  041056                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
5258  041034  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
5259  041042  012616                          MOV     (SP)+,(SP)
5260  041044  000002                          RTI                     ;;RETURN TO USER
5261  041046  023420                  $DTBL   10000
5262  041050  001750                          1000
5263  041052  000144                          100
5264  041054  000012                          10
5265  041056  000004                  $DBLK    BLKW   4
5266                                   SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5267
5268                                   ;;**********************************************************************
5269                                   ;*SAVE R0-R5
5270                                   ;*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

041066                                  $SAVREG
041066  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
041070  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
041072  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
041074  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
041076  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
041100  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
041102  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
041106  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
041112  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
041116  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
041122  000002                                  RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL
                                        ;*      RESREG
041124                                  $RESREG
041124  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
041130  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
041134  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
041140  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
041144  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
041146  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
041150  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
041152  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
041154  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
041156  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
041160  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

041162  104413                          $DB20   SAVREG                  ;;SAVE ALL REGISTERS
041164  016601  000002                          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
041170  012705  041301                          MOV     #$OCTVL+13,R5   ;;POINTER TO DATA TABLE
041174  012704  000014                          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
041200  012703  177770                          MOV     #^C7,R3         ;;MASK
```

```
5327  041204  012100            MOV     (R1)+,R0        ;;LOWER WORD
5328  041206  012101            MOV     (R1)+,R1        ;;HIGH WORD
5329  041210  005002            CLR     R2              ;;TERMINATOR
5330  041212  110245      1$    MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
5331  041214  010002            MOV     R0,R2           ;;GET THIS DIGIT
5332  041216  005304            DEC     R4              ;;COUNT THIS CHARACTER
5333  041220  003007            BGT     3$              ;;BR IF NOT THE LAST DIGIT
5334  041222  001405            BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
5335  041224  005205            INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5336  041226  010566  000002    MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
5337  041232  104414            RESREG                  ;;RESTORE ALL REGISTERS
5338  041234  000207            RTS     PC              ;;RETURN TO USER
5339  041236  006203      2$    ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
5340  041240  006001      3$    ROR     R1              ;;POSITION THE BINARY NUMBER FOR
5341  041242  006000            ROR     R0              ;;      THE NEXT OCTAL DIGIT
5342  041244  006001            ROR     R1
5343  041246  006000            ROR     R0
5344  041250  006001            ROR     R1
5345  041252  006000            ROR     R0
5346  041254  040302            BIC     R3,R2           ;;MASK OUT ALL JUNK
5347  041256  062702  000060    ADD     #'0,R2          ;;MAKE THIS CHAR  ASCII
5348  041262  000753            BR      1$              ;;GO PUT IT IN THE DATA TABLE
5349  041264  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
                                ;
041302  010046                  $TRAP   MOV     R0,-(SP)        ;;SAVE R0
041304  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
041310  005740                          TST     -(R0)           ;;BACKUP BY 2
041312  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
041314  006300                          ASL     R0              ;;POSITION FOR INDEXING
041316  016000  041336                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
041322  000200                          RTS     R0              ;;GO TO ROUTINE
                                ;
                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
                                ;
041324  011646                  $TRAP2  MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
041326  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
041334  000002                          RTI                     ;;RESTORE THE PSW
                                ;
                                 SBTTL  TRAP TABLE
                                ;
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION
                                ;
                                ;       ROUTINE
                                ;       -------
041336  041324                  $TRPAD   WORD   $TRAP2
041340  037610                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
041342  040440                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
041344  040414                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
041346  040454                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
041350  040642                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
041352  040340                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
                                ;
041354  036536                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
                                ;
041356  036466                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
041360  037010                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
041362  037130                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
041364  041066                          $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
041366  041124                          $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
                                 SBTTL  POWER DOWN AND UP ROUTINES
                                ;
                                ;;*************************************************************
                                ;POWER DOWN ROUTINE
041370  012737  041546  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
041376  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO-7
041404  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
041406  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
041410  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
041412  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
041414  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
```

```
5406  041416  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
5407  041420  017746  137514                  MOV     @SWR,-(SP)        ;;PUSH @SWR ON STACK
5408  041424  010637  041552                  MOV     SP,$SAVR6         ;;SAVE SP
5409  041430  012737  041442  000024          MOV     #$PWRUP,@#PWRVEC  ;;SET UP VECTOR
5410  041436  000000                          HALT
5411  041440  000776                          BR       -2               ;;HANG UP
5412
5413                                  ;;**********************************************************************
5414                                  ;POWER UP ROUTINE
5415  041442  012737  041546  000024  $PWRUP  MOV     #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
5416  041450  013706  041552                  MOV     $SAVR6,SP         ;;GET SP
5417  041454  005037  041552                  CLR     $SAVR6            ;;WAIT LOOP FOR THE TTY
5418  041460  005237  041552          1$      INC     $SAVR6            ;;WAIT FOR THE INC
5419  041464  001375                          BNE     1$                ;;OF  WORD
5420  041466  012677  137446                  MOV     (SP)+,@SWR        ;;POP STACK INTO @SWR
5421  041472  012605                          MOV     (SP)+,R5          ;;POP STACK INTO R5
5422  041474  012604                          MOV     (SP)+,R4          ;;POP STACK INTO R4
5423  041476  012603                          MOV     (SP)+,R3          ;;POP STACK INTO R3
5424  041500  012602                          MOV     (SP)+,R2          ;;POP STACK INTO R2
5425  041502  012601                          MOV     (SP)+,R1          ;;POP STACK INTO R1
5426  041504  012600                          MOV     (SP)+,R0          ;;POP STACK INTO R0
5427  041506  012737  041370  000024          MOV     #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
5428  041514  012737  000340  000026          MOV     #340,@#PWRVEC+2   ;;PRIO:7
5429  041522  104401                          TYPE                      ;;REPORT THE POWER FAILURE
5430  041524  041554                  $PWRMG   WORD    PWRMSG            ;;POWER FAIL MESSAGE POINTER
5431  041526  012716                          MOV     (PC)+,(SP)        ;;RESTART AT START
5432  041530  020000                  $PWRAD   WORD    START             ;;RESTART ADDRESS
5433  041532  042766  000020  000002          BIC     #20,2(SP)         ;;CLEAR "T" BIT
5434  041540  005037  035032                  CLR     $TBIT             ;;CLEAR THE "T" BIT FLAG
5435  041544  000002                          RTI
5436  041546  000000                  $ILLUP  HALT                      ;;THE POWER UP SEQUENCE WAS STARTED
5437  041550  000776                          BR       -2               ;; BEFORE THE POWER DOWN WAS COMPLETE
5438  041552  000000                  $SAVR6  0                         ;;PUT THE SP HERE
5439  041554  006412  050040  053517  PWRMSG   ASCIZ   <12><15>? POWER FAILURE - RESTART NG ?<12><15>
5440  041562  051105  043040  044501
5441  041570  052514  042522  026440
5442  041576  051040  051505  040524
5443  041604  052122  047111  020107
5444  041612  006412     000
5445          041616                   EVEN
5446
5447
```

```
                                                SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
5448
5449  041616  047125  054105  042520  EM1       ASCIZ  /UNEXPECTED CPU TRAP TO LOC  004/
5450  041624  052103  042105  044440
5451  041632  052520  052040  040522
5452  041640  020120  047524  046040
5453  041646  041517  020056  030060
5454  041654  000064
5455  041656  047125  054105  042520  EM2       ASCIZ  /UNEXPECTED MEM  MGMT  TRAP TO LOC  250/
5456  041664  052103  042105  046440
5457  041672  046505  020056  043515
5458  041700  052115  020056  051124
5459  041706  050101  052040  020117
5460  041714  047514  027103  031040
5461  041722  030065     000
5462  041725     120  044522  051117  EM3       ASCIZ  /PRIORITY BITS SET WRONG IN PSW/
5463  041732  052111  020131  044502
5464  041740  051524  051440  052105
5465  041746  053440  047522  043516
5466  041754  044440  020116  051520
5467  041762  000127
5468  041764  047515  042504  041040  EM4       ASCIZ  /MODE BITS SET WRONG IN PSW/
5469  041772  052111  020123  042523
5470  042000  020124  051127  047117
5471  042006  020107  047111  050040
5472  042014  063523     000
5473  042017     104  040525  020114  EM5       ASCIZ  /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5474  042024  042101  051104  051505
5475  042032  044523  043516  041040
5476  042040  062106  042527  047105
5477  042046  040040  023111  047514
5478  042054  041040  062131  051505
5479  042062  047440  020106  051520
5480  042070  000127
5481  042072  042613  047122  046105  EM6       ASCIZ  /KERNEL R6 CHANGED BY WRITING USER R6/
5482  042100  061040  020066  044103
5483  042106  047101  042507  020104
5484  042114  064602  053440  044522
5485  042122  044524  043516  052440
5486  042130  042523  020122  033122
5487  042136     000
5488  042137     101  046440  046505  EM7       ASCIZ  /A MEMORY MGMT  REG  TIMED OUT/
5489  042144  051117  020131  043515
5490  042152  052115  020056  042522
5491  042160  027107  062040  046511
5492  042166  042106  047440  052125
5493  042174     000
5494  042175     123  046525  040515  EM10      ASCIZ  /SUMMARY OF MEM  MGMT  REG  TIMEOUTS/
5495  042202  064522  047440  020106
5496  042210  042515  027115  046440
5497  042216  046507  027124  051040
5498  042224  043506  020056  044524
5499  042232  042515  052617  051524
5600  042240     000
5601  042241     115  046505  020056  EM11      ASCIZ  /MEM  MGMT  REG  WOULD NOT CLEAR/
5502  042246  043515  052115  020056
5503  042254  042522  027107  053440
```

```
5504  042262  052517  042114  047040
5505  042270  052117  041440  042514
5506  042276  051101     000
5507  042301     115  046505  020056  EM12    ASCIZ  /MEM. MGMT  REG  BITS NOT SET CORRECTLY/
5508  042306  043515  052115  020056
5509  042314  042522  027107  041040
5510  042322  052111  020123  047516
5511  042330  020124  042523  020124
5512  042336  047503  051122  041505
5513  042344  046124  000131
5514  042350  051123  020060  043105  EM13    ASCIZ  /SR0 EFFECTED BY WRITE TO PSW/
5515  042356  042506  052103  042105
5516  042364  041040  020131  051127
5517  042372  052111  020105  047524
5518  042400  060040  063523     000
5519  042406     123  030522  042040  EM14    ASCIZ  /SR1 DID NOT READ ALL ZEROS/
5520  042412  042111  047040  052117
5521  042420  051040  040505  020104
5522  042426  046101  020114  042532
5523  042434  047522  000123
5524  042440  052504  046101  040440  EM15    ASCIZ  /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5525  042446  042104  042522  051523
5526  042454  047111  020107  042502
5527  042462  053524  042505  020116
5528  042470  064502  042524  020123
5529  042476  043117  060040  051101
5530  042504  047440  020122  042120
5531  042512  000122
5532  042514  052504  046101  040440  EM16    ASCIZ  /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5533  042522  042104  042522  051523
5534  042530  047111  020107  042502
5535  042536  053524  042505  020116
5536  042544  040520  026522  042120
5537  042552  023522  000123
5538  042556  044120  051531  020056  EM17    ASCIZ  /PHYS  ADDR  FORMED WRONG IN MAINT  MODE/
5539  042564  042101  051104  020056
5540  042572  047506  046522  042105
5541  042600  053440  047522  043516
5542  042606  044440  020116  040515
5543  042614  047111  027124  046440
5544  042622  042117  000105
5545  042626  044120  051531  020056  EM20    ASCIZ  /PHYS  ADDR  FORMED WRONG IN RELOCATE MODE/
5546  042634  042101  051104  020056
5547  042642  047506  046522  042105
5548  042650  053440  047522  043516
5549  042656  044440  020116  042522
5550  042664  047514  040503  042524
5551  042672  046440  042117  000105
5552  042700  026527  044502  020124  EM21    ASCIZ  /W-BIT DID NOT GET SET IN PDR/
5553  042706  044504  020104  047516
5554  042714  020124  042507  020124
5555  042722  042523  020124  047111
5556  042730  060040  051104     000
5557  042736     127  041066  052111  EM22   .ASCIZ  /W-BIT SET IN MORE THAN ONE PDR/
5558  042742  051440  052105  044440
5559  042750  020116  047515  042522
```

```
5560  043756  052040  040510  020116
5561  043764  047117  020105  042120
5562  043772  000122
5563  043774  026527  044502  020124  EM23    ASCIZ  /W-BIT NOT CLEARED BY WRITING TO PDR/
5564  044002  047516  020124  046103
5565  044010  040606  042522  020104
5566  044016  064502  053440  044522
5567  044024  044524  043516  052040
5568  044032  020117  042120  000122
5569  044040  051127  052111  047111  EM24    ASCIZ  /WRITING SR0 SET W-BIT IN KIPDR7/
5570  044046  020107  051123  020060
5571  044054  042523  020124  026527
5572  044062  044502  020124  047111
5573  044070  043440  050111  051104
5574  044076  000067
5575  044100  026527  044502  020124  EM25    ASCIZ  /W-BIT GOT SET DURING ODD ADDR  ABORT/
5576  044106  047507  020124  042523
5577  044114  020124  052504  044522
5578  044122  043516  047440  042104
5579  044130  040440  042104  027122
5580  044136  040440  047502  052122
5581  044144     000
5582  044146     115  046505  051117  EM26    ASCIZ  /MEMORY MGMT  ACCESS ABORT DID NOT OCCUR/
5583  044152  020131  043515  052115
5584  044160  020066  041501  042503
5585  044166  051523  040440  047502
5586  044174  052122  042040  042111
5587  044202  047040  052117  047440
5588  044210  041503  051125     000
5589  044216     101  041503  051505  EM27    ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5590  044222  020123  051105  047522
5591  044230  020122  044504  020104
5592  044236  047516  020124  041101
5593  044244  051117  020124  047111
5594  044252  052123  052522  052103
5595  044260  047511  000116
5596  044264  051123  020060  044504  EM30    ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
5597  044272  020104  047516  020124
5598  044300  042522  047520  052122
5599  044306  040440  041503  051505
5600  044314  020123  051105  047522
5601  044322  020122  047503  051122
5602  044330  041505  046124  000131
5603  044336  044504  020104  047516  EM31.   ASCIZ  /DID NOT LOCKUP CORRECT VIRTUAL ADDR /
5604  044344  020124  047514  045503
5605  044352  050105  041440  051117
5606  044360  042522  052103  053040
5607  044366  051111  052524  046101
5608  044374  040440  042104  027122
5609  044402     000
5610  044403     120  043501  020105  EM32.   ASCIZ  /PAGE LGTH  ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5611  044410  043514  044124  020056
5612  044416  041101  051117  020124
5613  044424  041517  052503  051122
5614  044432  042105  053440  042510
5615  044440  020116  052111  051440
```

```
5616  043446  047510  046125  047104
5617  043454  052047  044040  053101
5618  043462  000105
5619  043464  040520  042507  046040  EM33    ASCIZ  /PAGE LGTH  ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5620  043472  052107  027110  040440
5621  043500  047502  052122  042040
5622  043506  042111  047040  052117
5623  043514  047440  041503  051125
5624  043522  053440  042510  020116
5625  043530  052111  051440  047510
5626  043536  046125  020104  040510
5627  043544  042526     000
5628  043547     123  030122  042040  EM34    ASCIZ  /SRO DID NOT REPORT PAGE LGTH  ABORT CORRECTLY/
5629  043564  042111  047040  052117
5630  043562  051040  050106  051117
5631  043570  020124  040520  042507
5632  043576  046040  052107  027110
5633  043604  040440  047502  052122
5634  043612  041440  051117  042522
5635  043620  052103  054514     000
5636  043625     123  030122  047440  EM37    ASCIZ  /SRO OR SR2 CHANGED BY A SECOND ABORT/
5637  043632  020122  051123  020062
5638  043640  044103  047101  042507
5639  043646  020104  054502  040440
5640  043654  051440  041505  047117
5641  043662  020104  041101  051117
5642  043670  000124
5643  043672  051123  020060  051117  EM40    ASCIZ  /SRO OR SR2 WERE NOT "RESET" BY A RESET/
5644  043700  051440  031122  053440
5645  043706  051105  020105  047516
5646  043714  020124  051042  051505
5647  043722  052105  020042  054502
5648  043730  040440  051040  051505
5649  043736  052105     000
5660  043741     123  031122  047040  EM41.   .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
5661  043746  052117  052040  040522
5662  043754  046603  047111  020107
5663  043762  047523  051122  041505
5664  043770  046124  000131
5665  043774  044604  020104  047516  EM42:   ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
5666  044002  020124  051124  020101
5667  044010  052040  051110  020125
5668  044016  042513  047122  046105
5669  044024  051440  040520  042503
5660  044032     000
5661  044033     113  020124  051105  EM43:   .ASCIZ  /KT ERROR SERVICED ON ODD ADDR  ERROR/
5662  044040  047522  020122  042523
5663  044046  053122  041511  042105
5664  044054  047440  020116  042117
5665  044062  020104  042101  051104
5666  044070  020066  051106  047522
5667  044076  000122
5668  044100  051123  020060  051117  EM44:   ASCIZ  /SRO OR SR2 CHANGED BY ODD ADDR  ERROR/
5669  044106  051440  031122  041440
5670  044114  040610  043516  042105
5671  044122  041040  020131  042117
```

```
5672  044130  020104  042101  051104
5673  044136  020056  051105  047522
5674  044144  000122
5675  044146  051105  047522  020122  EM45    ASCIZ  /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR )/
5676  044154  052504  044522  043516
5677  044162  021040  047504  041125
5678  044170  042514  042440  051122
5679  044176  051117  020042  045450
5680  044204  020124  020046  042117
5681  044212  020104  042101  051104
5682  044220  024456     000
5683  044223     115  060106  020111  EM46    ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
5684  044230  047111  052123  052522
5685  044236  052103  047511  020116
5686  044244  052520  044123  042105
5687  044252  053440  047522  043516
5688  044260  042040  052101  000101
5689  044266  052116  044520  044440  EM47    ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
5690  044274  051516  051124  041525
5691  044302  044524  047117  046040
5692  044310  040617  042504  020104
5693  044316  051127  047117  020107
5694  044324  040604  040524     000
5695  044331     123  040524  045503  EM50    ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
5696  044336  047040  052117  050040
5697  044344  051525  042510  020104
5698  044352  054502  046440  050106
5699  044360  026511  052115  044520
5700  044366     000
5701  044367     113  051105  042516  EM51    ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER  MFPI-MTPI/
5702  044374  020114  040520  042507
5703  044402  040440  041503  051505
5704  044410  020123  047111  052123
5705  044416  040506  020104  043117
5706  044424  052440  042523  035122
5707  044432  046440  050106  026511
5708  044440  052116  044520     000
5709  044446     127  047522  043516  EM52    ASCIZ  /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5710  044462  060040  051104  051447
5711  044460  051040  043106  051105
5712  044466  047106  042503  020104
5713  044474  044127  046111  020105
5714  044502  047111  051040  046105
5715  044510  041517  052101  020105
5716  044516  047516  042504     000
5717  044523     116  060106  020104  EM53    ASCIZ  /MFPD INSTRUCTION PUSHED WRONG DATA/
5718  044530  047111  052123  052522
5719  044536  052103  047511  020116
5720  044544  052520  044123  042105
5721  044552  053440  047522  043516
5722  044560  042040  052101  000101
5723  044566  052123  041501  020113  EM54    ASCIZ  /STACK NOT PUSHED BY MFPD-MTPD/
5724  044574  047516  020124  052520
5725  044602  044123  042105  041040
5726  044610  020131  043115  042120
5727  044616  046456  060124  000104
```

```
5728  044624  040520  020122  051117  EM55    ASCIZ  /PAR OR PDR CHANGED BY A RESET/
5729  044632  050040  051104  041440
5730  044640  040510  043516  042105
5731  044646  041040  020131  020101
5732  044654  042522  042523  000124
5733  044662  046111  042514  040507  EM56    ASCIZ  /ILLEGAL MODE 01 NOT ABORTED/
5734  044670  020114  047515  042504
5735  044676  030040  020061  047516
5736  044704  020124  041101  051117
5737  044712  042524  000104
5738  044716  051123  020060  044504  EM57    ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 01 CORRECTLY/
5739  044724  020104  047516  020124
5740  044732  042522  047520  052122
5741  044740  044440  046114  043505
5742  044746  046101  046440  042117
5743  044754  020106  030460  041440
5744  044762  051117  042522  052103
5745  044770  064514      000
5746  044773      120  053523  041440  EM60    ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
5747  045000  040610  043516  042105
5748  045006  041040  020131  047101
5749  045014  051040  044524  044440
5750  045022  020116  051525  051105
5751  045030  046440  042117  000105
5752  045036  040615  047111  027124  EM61    ASCIZ  /MAINT  MODE (SR0 <8>) NOT DISABLED BY A RESET/
5753  045044  046440  042117  020105
5754  045052  051460  030122  036040
5756  045060  037070  020061  047516
5756  045066  020124  044504  040523
5757  045074  046102  042106  041040
5758  045102  020131  020101  042522
5759  045110  042523  000124
5760  045114  040604  040624  044440  EM62    ASCIZ  /DATA INCORRECT AFTER A MAINT  MODE WRITE/
5761  045122  041516  051117  042522
5762  045130  052103  040440  052106
5763  045136  051105  040440  046440
5764  045144  044601  052116  020056
5766  045152  047515  042504  053440
5766  045160  044522  042524      000
5767  045165      123  052517  041522  EM63    ASCIZ  /SOURCE RELOCATED IN MAINT  MODE/
5768  045172  020105  042522  047514
5769  045200  040603  042524  020104
5770  045206  047111  046440  044501
5771  045214  052116  020056  047515
5772  045222  042504      000
5773  045225      116  047117  051040  EM64    ASCIZ  /NON RESIDENT ABORT DID NOT OCCUR/
5774  045232  051505  042111  047105
5775  045240  020124  041101  051117
5776  045246  020124  044504  020104
5777  045254  047516  020124  041517
5778  045262  052503  000122
5779  045266  051105  047522  020122  EM65    ASCIZ  /ERROR FLAG FOR NR ABORT (BIT15) IN SR0 DID NOT SET/
5780  045274  046105  043501  043040
5781  045302  051117  047040  020122
5782  045310  041101  051117  020124
5783  045316  041050  052111  032461
```

```
5784  046324  020051  047111  051440
5785  046332  030122  042040  042111
5786  046340  047040  052117  051440
5787  046346  052105     000
5788  046351     123  031122  042040  EM66    ASCIZ  /SR2 DID NOT FREEZE THE VIRTUAL ADDRS OF THE ABORTD INTR/
5789  046366  042111  047040  052117
5790  046364  043040  042522  055105
5791  046372  020105  044124  020105
5792  046400  044526  052122  040525
5793  046406  020114  042101  051104
5794  046414  020123  043117  052040
5795  046422  042510  040440  047502
5796  046430  062122  020104  047111
5797  046436  061124     000
5798  046441     062  042116  047040  EM67    ASCIZ  /2ND NON RESIDENT ABORT DID NOT OCCUR/
5799  046446  047117  051040  051505
5800  046454  042111  047105  020124
5801  046462  041101  051117  020124
5802  046470  044504  020104  047516
5803  046476  020124  041517  052503
5804  046504  000122
5805  046506  047062  020104  047516  EM70    ASCIZ  /2ND NON RESIDENT ABORT CAUSED SR2 TO CHANGE/
5806  046514  020116  042522  044523
5807  046522  042504  052116  040440
5808  046530  047502  052122  041440
5809  046536  062501  042523  020104
5810  046544  051123  020062  047524
5811  046562  041440  040510  043516
5812  046560  000106
5813  046562  061123  020060  040527  EM71    ASCIZ  /SR0 WAS NOT CLEARED BY INIT  /
5814  046570  020123  047516  020124
5815  046576  046103  040505  042522
5816  046604  020104  064502  044440
5817  046612  044516  027124  000040
5818
5819  046620  046117  020104  041520  DH1     ASCIZ  /OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC/
5820  046626  020040  046117  020104
5821  046634  051520  020127  033122
5822  046642  053440  051501  020040
5823  046650  042524  052123  047516
5824  046656  020040  051105  047522
5825  046664  050122  000103
5826  046670  046117  020104  041520  DH2     ASCIZ  /OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC/
5827  046676  020040  046117  020104
5828  046704  051520  020127  033122
5829  046712  053440  051501  020040
5830  046720  051123  020060  020040
5831  046726  020040  051123  020062
5832  046734  020040  020040  042524
5833  046742  052123  047516  020040
5834  046750  051105  047522  050122
5835  046756  000103
5836  046760  051127  052117  020105  DH3     ASCIZ  /WROTE   READ    TESTNO  ERRORPC/
5837  046766  020040  042522  042101
5838  046774  020040  020040  042524
5839  047002  052123  047516  020040
```

```
5840  046010  051105  047522  050122
5841  046016  000103
5842  046020  042101  051104  051505  DH7     ASCIZ  /ADDRESS TESTNO  ERRORPC/
5843  046026  020123  042524  052123
5844  046034  047516  020040  051105
5845  046042  047522  050122  000103
5846  046050  042522  044507  052123  DH10    ASCII  /REGISTER-ADDRS  NUM  OF/<CRLF>
5847  046056  051105  040455  042104
5848  046064  051522  020040  052516
5849  046072  020115  047440  100106
5850  046100  047101  026504  042105          ASCIZ  /AND-ED  OR-ED   TIMOUTS TESTNO  ERRORPC/
5851  046106  020040  061117  042455
5852  046114  020104  020040  044524
5853  046122  047515  052125  020123
5854  046130  042524  052123  047516
5855  046136  020040  061105  047522
5856  046144  060122  000103
5857  046150  042522  044507  052123  DH11    ASCII  /REGISTR READ    READ-(BINARY)/<CRLF>
5858  046156  020122  042522  042101
5859  046164  020040  020040  042522
5860  046172  042101  024066  044502
5861  046200  040616  064522  100051
5862  046206  042101  061104  051505          ASCIZ  /ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC/
5863  046214  020123  047450  052103
5864  046222  046101  020061  032065
5865  046230  031063  030061  034071
5866  046236  033067  032066  031063
5867  046244  030061  020040  042524
5868  046252  052123  047516  020040
5869  046260  061105  047522  050122
5870  046266  000103
5871  046270  042522  044507  052123  DH12    ASCII  /REGISTR WROTE   READ    READ-(BINARY)/<CRLF>
5872  046276  020122  061127  052117
5873  046304  020106  020040  042522
5874  046312  042101  020040  020040
5875  046320  042522  042101  024055
5876  046326  044502  040516  054522
5877  046334  100051
5878  046336  042101  061104  051505          ASCIZ  /ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC/
5879  046344  020123  047450  052103
5880  046352  046101  020061  047450
5881  046360  052103  046101  020051
5882  046366  032066  031063  030061
5883  046374  034071  033067  032065
5884  046402  031063  030061  020040
5885  046410  042524  052123  047516
5886  046416  020040  061105  047522
5887  046424  060122  000103
5888  046430  042522  042101  020040  DH13    ASCIZ  /READ    TESTNO  ERRORPC/
5889  046436  020040  042524  052123
5890  046444  047516  020040  051105
5891  046452  047522  060122  000103
5892  046460  040620  026522  042120  DH16    ASCII  /PAR-PDR PAR-PDR/<CRLF>
5893  046466  020122  040520  026522
5894  046474  042120  100122
5895  046500  046103  040505  042522          ASCIZ  /CLEARED EFFECTD EXPECTD RECEIVD TESTNO  ERRORPC/
```

```
5896  046506  020104  043105  042506
5897  046514  052103  020104  054105
5898  046522  042520  052103  020104
5899  046530  042522  042503  053111
5900  046536  020104  042524  052123
5901  046544  047516  020040  051105
5902  046552  047522  050122  000103
5903  046560  044120  051531  041511  DH17    ASCII  /PHYSICL VIRTUAL/<CRLF>
5904  046566  020114  044526  052122
5905  046574  040525  100114
5906  046600  042101  051104  051505          ASCIZ  /ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC/
5907  046606  020123  042101  051104
5908  046614  051505  020123  044513
5909  046622  040620  032122  020040
5910  046630  042524  052123  047516
5911  046636  020040  051105  047522
5912  046644  050122  000103
5913  046650  044120  051531  041511  DH20    ASCII  /PHYSICL PAR 4   PAR 5/<CRLF>
5914  046656  020114  040520  020122
5915  046664  020064  020040  040520
5916  046672  020122  100065
5917  046676  042101  051104  051505          ASCIZ  /ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO  ERRORPC/
5918  046704  020123  041126  020101
5919  046712  020040  020040  041126
5920  046720  020101  020040  020040
5921  046726  040620  020122  020064
5922  046734  020040  040620  020122
5923  046742  020066  020040  051520
5924  046750  020127  020040  020040
5925  046756  042524  052123  047516
5926  046764  020040  051105  047522
5927  046772  050122  000103
5928  046776  042120  020122  020040  DH21    ASCII  /PDR     VIRTUAL/<CRLF>
5929  047004  020040  044526  052122
5930  047012  040625  100114
5931  047016  042524  052123  042105          ASCIZ  /TESTED  ADDRESS TESTNO  ERRORPC/
5932  047024  020040  042101  051104
5933  047032  051505  020123  042524
5934  047040  052123  047516  020040
5935  047046  051105  047522  050122
5936  047054  000103
5937  047056  042120  020122  047111  DH22    ASCII  /PDR IN  PDR     VIRTUAL/
5938  047064  020040  042120  020122
5939  047072  020040  020040  044526
5940  047100  052122  040625     114
5941  047106     105  051122  051117          ASCIZ  /ERROR   TESTED  ADDRESS TESTNO  ERRORPC/
5942  047112  020040  052040  051505
5943  047120  042524  020104  040440
5944  047126  042104  042622  051523
5945  047134  052040  051505  047124
5946  047142  020117  042440  051122
5947  047150  051117  041520     000
5948  047156     120  051104  020040  DH23    ASCIZ  /PDR     TESTNO  ERRORPC/
5949  047162  020040  052040  051505
5950  047170  047124  020117  042440
5951  047176  051122  051117  041520
```

```
5952  047204      000
5953  047205      120  051104  053440  DH24    ASCIZ  /PDR WAS EXPECTD TESTNO  ERRORPC/
5954  047212  051501   042440  050130
5955  047220  041505   042124  052040
5956  047226  051505   047124  020117
5957  047234  042440   051122  051117
5958  047242  041520      000
5959  047245      120  051104  032040  DH26    ASCIZ  /PDR 4   PSW     TESTNO  ERRORPC/
5960  047252  020040   050040  053523
5961  047260  020040   020040  052040
5962  047266  051505   047124  020117
5963  047274  042440   051122  051117
5964  047302  041520      000
5965  047305      123  030122  053440  DH30    ASCIZ  /SR0 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
5966  047312  051501   042440  050130
5967  047320  041505   042124  050040
5968  047326  051104   032040  020040
5969  047334  060040   053523  020040
5970  047342  020040   052040  051505
5971  047350  047124   020117  042440
5972  047356  051122   051117  041520
5973  047364      000
5974  047365      123  031122  053440  DH31    ASCIZ  /SR2 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
5975  047372  051501   042440  050130
5976  047400  041505   042124  050040
5977  047406  051104   032040  020040
5978  047414  060040   053523  020040
5979  047422  020040   052040  051505
5980  047430  047124   020117  042440
5981  047436  051122   051117  041520
5982  047444      000
5983  047445      126  041056  040456  DH32    ASCIZ  /V B A   KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC/
5984  047452  020056   045440  050111
5985  047460  051104   020064  051440
5986  047466  030122   053440  051501
5987  047474  051440   031122  053440
5988  047502  051501   052040  051505
5989  047510  047124   020117  042440
5990  047516  051122   051117  041520
5991  047524      000
5992  047525      126  041056  040456  DH33    ASCIZ  /V B A   KIPDR4  TESTNO  ERRORPC/
5993  047532  020056   045440  050111
5994  047540  051104   020064  052040
5995  047546  051505   047124  020117
5996  047554  042440   051122  051117
5997  047562  041520      000
5998  047565      126  041056  040456  DH34    ASCIZ  /V B A   KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC/
5999  047572  020056   045440  050111
6000  047600  051104   020064  051440
6001  047606  030122   053440  051501
6002  047614  042440   050130  041505
6003  047622  042124   052040  051505
6004  047630  047124   020117  042440
6005  047636  051122   051117  041520
6006  047644      000
6007  047645      126  041056  040456  DH35    ASCIZ  /V B A   KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC/
```

```
 6008  047652  020056  045440  050111
 6009  047660  051104  020064  051440
 6010  047666  031122  053440  051501
 6011  047674  042440  050130  041505
 6012  047702  042124  052040  051505
 6013  047710  047124  020117  042440
 6014  047716  051122  051117  041520
 6015  047724     000
 6016  047725     123  031122  053440  DH36    ASCIZ  /SR2 WAS EXPECTD TESTNO  ERRORPC/
 6017  047732  051501  042440  050130
 6018  047740  041505  042124  052040
 6019  047746  051505  047124  020117
 6020  047754  042440  051122  051117
 6021  047762  041520     000
 6022  047765     106  051111  052123  DH37    ASCI I /FIRST ABORT     SECOND ABORT/<CRLF>
 6023  047772  040440  047502  052122
 6024  050000  020040  020040  051440
 6025  050006  041505  047117  020104
 6026  050014  041101  051117  100124
 6027  050022  051123  020060  040527          ASCIZ  /SRO WAS SP2 WAS SRO WAS SR2 WAS TESTNO  ERRORPC/
 6028  050030  020123  051123  020062
 6029  050036  040527  020123  051123
 6030  050044  020060  040527  020123
 6031  050052  051123  020062  040527
 6032  050060  020123  042524  052123
 6033  050066  047516  020040  051105
 6034  050074  047522  050122  000103
 6035  050102  051123  020060  040527  DH40    ASCIZ  /SRO WAS SR2 WAS TESTNO  ERRORPC/
 6036  050110  020123  051123  020062
 6037  050116  040527  020123  042524
 6038  050124  052123  047516  020040
 6039  050132  051105  047522  050122
 6040  050140  000103
 6041  050142  051520  020127  040527  DH42    ASCIZ  /PSW WAS R6 WAS  TESTNO  ERRORPC/
 6042  050150  020123  033122  053440
 6043  050156  051501  020040  042524
 6044  050164  052123  047516  020040
 6045  050172  051105  047522  050122
 6046  050200  000103
 6047  050202  054105  042520  052103  DH44    ASCII  /EXPECTED         RECEIVED/<CRLF>
 6048  050210  042105  020040  020040
 6049  050216  020040  020040  051040
 6050  050224  041505  044505  042526
 6051  050232  100104
 6052  050234  051123  020060  020040          ASCIZ  /SRO     SR2     SRO WAS SR2 WAS TESTNO  ERRORPC/
 6053  050242  020040  051123  020062
 6054  050250  020040  020040  051123
 6055  050256  020060  040527  020123
 6056  050264  051123  020062  040527
 6057  050272  020123  042524  052123
 6058  050300  047516  020040  051105
 6059  050306  047522  050122  000103
 6060  050314  054105  042520  052103  DH45    ASCII  /EXPECTED /<CRLF>
 6061  050322  042105  100072
 6062  050326  051520  020127  020040          ASCII  /PSW     PC      SRO     SR2/<CRLF>
 6063  050334  020040  041520  020040
```

```
6064  050342  020040  020040  051123
6065  050350  020060  020040  020040
6066  050356  051123  100062
6067  050362  033461  030060  033461          ASCII  /170017  (3$+4)  020147  (3$)/<CRLF>
6068  050370  020040  031450  025444
6069  050376  024464  020040  031060
6070  050404  030460  033464  020040
6071  050412  031450  024444     200
6072  050417     122  041505  044505          ASCII  /RECEIVED /<CRLF>
6073  050424  042526  035104     200
6074  050431     120  053523  020040          ASCIZ  /PSW     PC      SR0     SR2     TESTNO  ERRORPC/
6075  050436  020040  050040  020103
6076  050444  020040  020040  051440
6077  050452  030122  020040  020040
6078  050460  051440  031122  020040
6079  050466  020040  052040  051505
6080  050474  047124  020117  042440
6081  050502  051122  051117  041520
6082  050510     000
6083  050511     104  052101  020101  DH46    ASCII  /DATA    DATA/<CRLF>
6084  050516  020040  042040  052101
6085  050524  100101
6086  050526  064105  042520  052103          ASCIZ  /EXPECTD RECEIVD TESTNO  ERRORPC/
6087  050534  020104  042522  042503
6088  050542  063111  020104  042524
6089  050550  052123  047516  020040
6090  050556  051105  047522  050122
6091  050564  000103
6092  050566  042524  052123  047516  DH50    ASCIZ  /TESTNO  ERRORPC/
6093  050574  020040  051105  047522
6094  050602  050122  000103
6095  050606  051123  020060  040527  DH51    ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
6096  050614  020123  051123  020062
6097  050622  040527  020123  042524
6098  050630  052123  047516  020040
6099  050636  051105  047522  050122
6100  050644  000103
6101  050646  044120  051531  041511  DH52    ASCII  /PHYSICL PAR 4/<CRLF>
6102  050654  020114  040520  020122
6103  050662  100064
6104  050664  042101  051104  051505          ASCIZ  /ADDRESS V B A   PAR 4   SR0 WAS SR2 WAS PSW     TESTNO  ERRORPC/
6105  050672  020123  027126  027102
6106  050700  027101  020040  040520
6107  050706  020122  020064  020040
6108  050714  051123  020060  040527
6109  050722  020123  051123  020062
6110  050730  040527  020123  051520
6111  050736  020127  020040  020040
6112  050744  042524  052123  047516
6113  050752  020040  051105  047522
6114  050760  050122  000103
6115  050764  051123  020060  040527  DH57    ASCIZ  /SR0 WAS EXPECTD TESTNO  ERRORPC/
6116  050772  020123  054105  042520
6117  051000  052103  020104  042524
6118  051006  052123  047516  020040
6119  051014  051105  047522  050122
```

```
6120  051022  000103
6121  051024  051520  020127  040527  DH60    ASCIZ  /PSW WAS EXPECTD TESTNO  ERRORPC/
6122  051032  020123  054105  042520
6123  051040  052103  020104  042524
6124  051046  052123  047516  020040
6125  051054  051105  047522  050122
6126  051062  000103
6127  051064  051123  020060  020040  DH61    ASCIZ  /SR0     SR2   TESTNO  ERRORPC/
6128  051072  020040  051123  020062
6129  051100  020040  042524  052123
6130  051106  047516  020040  051105
6131  051114  047522  050122  000103
6132  051122  051123  020062  054105  DH62    ASCIZ  /SR2 EXP SR2 REC TESTNO ERRORPC/
6133  051130  020120  051123  020062
6134  051136  042522  020103  042524
6135  051144  052123  047516  042440
6136  051152  051122  051117  041520
6137  051160     000
6138  051161     123  030122  042440  DH63    ASCIZ  /SR0 EXP SR0 REC TESTNO ERRORPC/
6139  051166  050130  051440  030122
6140  051174  051040  041505  052040
6141  051202  051505  047124  020117
6142  051210  051105  047522  050122
6143  051216  000103
6144
6145                                  EVEN
6146
6147  051220  001356  001360  001354  DT1     WORD   TRAPPC, TRAPPS, WASR6, TESTNO, SERRPC, 0
6148  051226  001352  001116  000000
6149  051234  001356  001360  001354  DT2     WORD   TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, SERRPC, 0
6150  051242  001366  001370  001352
6151  051250  001116  000000
6152  051254  001162  001164  001352  DT3     WORD   $REG0, $REG1, TESTNO, SERRPC, 0
6153  051262  001116  000000
6154  051266  001162  001352  001116  DT7     WORD   $REG0, TESTNO, SERRPC, 0
6155  051274  000000
6156  051276  001374  001376  001400  DT10    WORD   ANDADR, ORADR, TONUM, TESTNO, SERRPC, 0
6157  051304  001352  001116  000000
6158  051312  001162  001164  001164  DT11    WORD   $REG0, $REG1, $REG1, TESTNO, SERRPC, 0
6159  051320  001352  001116  000000
6160  051326  001162  001164  001166  DT12    WORD   $REG0, $REG1, $REG2, $REG2, TESTNO, SERRPC, 0
6161  051334  001166  001352  001116
6162  051342  000000
6163  051344  001162  001352  001116  DT13    WORD   $REG0, TESTNO, SERRPC, 0
6164  051352  000000
6165  051354  001162  001164  001174  DT16    WORD   $REG0, $REG1, $REG5, $REG2, TESTNO, SERRPC, 0
6166  051362  001166  001352  001116
6167  051370  000000
6168  051372  001406  001402  001172  DT17    WORD   PBALO, VIRT1, $REG4, TESTNO, SERRPC, 0
6169  051400  001352  001116  000000
6170  051406  001406  001402  001404  DT20    WORD   PBALO, VIRT1, VIRT2, $REG4, $REG5, STMP0, TESTNO, SERRPC, 0
6171  051414  001172  001174  001176
6172  051422  001352  001116  000000
6173  051430  001174  001170  001352  DT21    WORD   $REG5, $REG3, TESTNO, SERRPC, 0
6174  051436  001116  000000
6175  051442  001162  001174  001170  DT22    WORD   $REG0, $REG5, $REG3, TESTNO, SERRPC, 0
```

```
 6176  051450  001352  001116  000000
 6177  051456  001174  001352  001116  DT23    WORD    $REG5,TESTNO,$ERRPC,0
 6178  051464  000000
 6179  051466  001166  001164  001352  DT24    WORD    $REG2,$REG1,TESTNO,$ERRPC,0
 6180  051474  001116  000000
 6181  051500  001166  001176  001352  DT26    WORD    $REG2,$TMP0,TESTNO,$ERRPC,0
 6182  051506  001116  000000
 6183  051512  001366  001170  001166  DT30    WORD    WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
 6184  051520  001176  001352  001116
 6185  051526  000000
 6186  051530  001370  001172  001166  DT31    WORD    WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
 6187  051536  001176  001352  001116
 6188  051544  000000
 6189  051546  001162  001172  001366  DT32    WORD    $REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6190  051564  001370  001352  001116
 6191  051562  000000
 6192  051564  001162  001172  001352  DT33    WORD    $REG0,$REG4,TESTNO,$ERRPC,0
 6193  051572  001116  000000
 6194  051576  001162  001172  001366  DT34    WORD    $REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
 6195  051604  001166  001352  001116
 6196  051612  000000
 6197  051614  001162  001172  001370  DT35    WORD    $REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
 6198  051622  001170  001352  001116
 6199  051630  000000
 6200  051632  001370  001164  001352  DT36    WORD    WASSR2,$REG1,TESTNO,$ERRPC,0
 6201  051640  001116  000000
 6202  051644  001176  001202  001366  DT37    WORD    $TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6203  051652  001370  001352  001116
 6204  051650  000000
 6205  051652  001366  001370  001352  DT40    WORD    WASSR0,WASSR2,TESTNO,$ERRPC,0
 6206  051670  001116  000000
 6207  051674  001164  001166  001352  DT42    WORD    $REG1,$REG2,TESTNO,$ERRPC,0
 6208  051702  001116  000000
 6209  051706  001162  001164  001366  DT44    WORD    $REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6210  051714  001370  001352  001116
 6211  051722  000000
 6212  051724  001164  001170  001366  DT45.   WORD    $REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
 6213  051732  001370  001352  001116
 6214  051740  000000
 6215  051742  001162  001164  001352  DT46    WORD    $REG0,$REG1,TESTNO,$ERRPC,0
 6216  051750  001116  000000
 6217  051754  001352  001116  000000  DT50.   WORD    TESTNO,$ERRPC,0
 6218  051762  001366  001370  001352  DT51.   WORD    WASSR0,WASSR2,TESTNO,$ERRPC,0
 6219  051770  001116  000000
 6220  051774  001406  001402  001172  DT52.   WORD    PBFLO,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
 6221  052002  001366  001370  001176
 6222  052010  001352  001116  000000
 6223  052016  001366  001164  001352  DT57:   WORD    WASSR0,$REG1,TESTNO,$ERRPC,0
 6224  052024  001116  000000
 6225  052030  001164  001166  001352  DT60.   WORD    $REG1,$REG2,TESTNO,$ERRPC,0
 6226  052036  001116  000000
 6227  052042  001370  001172  001352  DT64    WORD    WASSR2,$REG4,TESTNO,$ERRPC,0
 6228  052060  001116  000000
 6229  052064  001366  001370  001352  DT65.   WORD    WASSR0,WASSR2,TESTNO,$ERRPC,0
 6230  052062  001116  000000
 6231  052066  001364  001370  001352  DT66    WORD    CORSR2,WASSR2,TESTNO,$ERRPC,0
```

```
 6232  052074  001116  000000
 6233  052100  001362  001366  001352  DT67     WORD   CORSRO,WASSRO,TESTNO,SERRPC,0
 6234  052106  001116  000000
 6235
 6236  052112     000     000     000  DF1      BYTE   0,0,0,0,0
 6237  052115     000     000
 6238  052117     000     000     000  DF2      BYTE   0,0,0,0,0,0,0
 6239  052122     000     000     000
 6240  052125     000
 6241  052126     000     000     000  DF3      BYTE   0,0,0,0
 6242  052131     000
 6243  052132     000     000     000  DF7      BYTE   0,0,0
 6244  052135     000     000     001  DF10     BYTE   0,0,1,0,0
 6245  052140     000     000
 6246  052142     000     000     002  DF11     BYTE   0,0,2,0,0
 6247  052145     000     000
 6248  052147     000     000     000  DF12     BYTE   0,0,0,2,0,0
 6249  052152     002     000     000
 6250  052155     000     000     000  DF13     BYTE   0,0,0
 6251  052160     000     000     000  DF16     BYTE   0,0,0,0,0,0
 6252  052163     000     000     000
 6253  052166     003     000     000  DF17.    BYTE   3,0,0,0,0
 6254  052171     000     000
 6255  052173     003     000     000  DF20     BYTE   3,0,0,0,0,0,0,0
 6256  052176     000     000     000
 6257  052201     000     000
 6258  052203     000     000     000  DF21     BYTE   0,0,0,0
 6259  052206     000
 6260  052207     000     000     000  DF22     BYTE   0,0,0,0,0
 6261  052212     000     000
 6262  052214     000     000     000  DF23     BYTE   0,0,0
 6263  052217     000     000     000  DF24     BYTE   0,0,0,0
 6264  052222     000
 6265  052223     000     000     000  DF30     BYTE   0,0,0,0,0,0
 6266  052226     000     000     000
 6267  052231     000     000     000  DF46     BYTE   0,0,0,0
 6268  052234     000
 6269  052235     000     000          DF50     BYTE   0,0
 6270  052237     000     000     000  DF51     BYTE   0,0,0,0
 6271  052242     000
 6272  052243     003     000     000  DF52     BYTE   3,0,0,0,0,0,0,0
 6273  052246     000     000     000
 6274  052251     000     000
 6275  052253     000     000     000  DF57     BYTE   0,0,0,0
 6276  052256     000
 6277  052257     000     000     000  DF60     BYTE   0,0,0,0
 6278  052262     000
 6279  052263     000     000     000  DF61     BYTE   0,0,0,0
 6280  052266     000
 6281
 6282          000001                           END
```

```
ABASE = 000000            815     856
ACDW1 = 000000            815     858
ACDW2 = 000000            815     859
ACPU0P= 000000            815     830
ADDW0 = 000000            815     860
ADDW1 = 000000            815     861
ADDW10= 000000            815     870
ADDW11= 000000            815     871
ADDW12= 000000            815     872
ADDW13= 000000            815     873
ADDW14= 000000            815     874
ADDW15= 000000            815     875
ADDW2 = 000000            815     862
ADDW3 = 000000            815     863
ADDW4 = 000000            815     864
ADDW5 = 000000            815     865
ADDW6 = 000000            815     866
ADDW7 = 000000            815     867
ADDW8 = 000000            815     868
ADDW9 = 000000            815     869
ADEVCT= 000000            815     821
ADEVM = 000000            815     857
AENW  = 000000            815     826
AENWM = 000000            815     827
AFATBL= 000000            815     818
AMADR1= 000000            815     843
AMADR2= 000000            815     847
AMADR3= 000000            815     850
AMADR4= 000000            815     853
AMAMS1= 000000            815     837
AMAMS2= 000000            815     845
AMAMS3= 000000            815     848
AMAMS4= 000000            815     851
AMSGRD= 000000            815     823
AMSGLB= 000000            815     824
AMSGTY= 000000            815     817
AMTYP1= 000000            815     838
AMTYP2= 000000            815     846
AMTYP3= 000000            815     849
AMTYP4= 000000            815     852
AMDMDR  001374            890#    1580*   1602*   1620*   1642*   1660*   1682*   1700*   1722*   1740*   1762*   6156
APARB = 000000            815     820
APRIOR= 000000            815
APTCSU= 000040           4992    5097#
APTDMW= 000001           4457    4985    5053    5095#
APTSIZ= 000200           1398    5094#
APTSP0= 000100           4987    5055    5096#
ASLAVE= 000000            815     828
ATESTM= 000000            815     819
AUNIT = 000000            815     822
AUSAR = 000000            815     829
AVECT1= 000000            815     854
AVECT2= 000000            815     855
BIT0  = 000001            628#
BIT00 = 000001            618#    628
BIT01 = 000002            617#    627
```

```
$IT02 = 000004          616#    626
$IT03 = 000010          615#    625
$IT04 = 000020          614#    624
$IT05 = 000040          613#    623
$IT06 = 000100          612#    622
$IT07 = 000200          611#    621
$IT08 = 000400          610#    620     4385
$IT09 = 001000          609#    619     4393    4468
$IT1  = 000002          627#
$IT10 = 002000          608#    4445
$IT11 = 004000          607#    4400
$IT12 = 010000          606#    4340
$IT13 = 020000          605#    4452
$IT14 = 040000          604#    4371
$IT15 = 100000          603#    1799
$IT2  = 000004          626#
$IT3  = 000010          625#
$IT4  = 000020          624#     702
$IT5  = 000040          623#
$IT6  = 000100          622#     703
$IT7  = 000200          621#
$IT8  = 000400          620#    2207    2209    2260    2278    2344    2370    2396    2423    2446    2488    2514    2540
                       2566    2592
$IT9  = 001000          619#
$PTVEC= 000014          635#
CKSUM = 104410         4370    4434    4467    5390#
CMPREG  036162         2078    2087    2096    2105    4649#
CMSG    037537         4942    4953#
CNTRLC  037460         4788    4876    4940#
CORSR0  001362          885#   3480#   6233
CORSR2  001364          886#   3468#   6231
CR    = 000015          543#    5031    5041
CRLF  = 000200          544#    1422    5002    5041    5846    5857    5871    5892    5903    5913    5928    6022    6047
                       6060    6062    6067    6072    6083    6101
DDISP = 177570          550#     782    1386
DF1     062112          917    6236#
DF10    062135          960    6244#
DF11    062142          967    1199    6246#
DF12    062147          974     993    6248#
DF13    062155          980     986    6250#
DF16    062160         1000    6251#
DF17    062166         1007    6253#
DF2     062117          923    1235    6238#
DF20    062173         1014    6255#
DF21    062203         1021    6258#
DF22    062207         1028    6260#
DF23    062214         1034    1130    6262#
DF24    062217         1040    1046    1052    1058    1082    1099    1112    1118    1124    1241    6263#
DF3     062126          929     935     941     947    1229    6241#
DF30    062223         1064    1070    1076    1088    1093    1106    1137    1147    6265#
DF46    062231         1154    1161    1186    6267#
DF50    062235         1167    1192    1205    1223    6269#
DF51    062237         1173    6270#
DF52    062243         1180    6272#
DF57    062253         1211    6275#
DF60    062257         1217    6277#
```

```
DF61    052263          1247    1253    1260    1267    1273    1279    6279#
DF7     052132           953    6243#
DH1     045620           915    5819#
DH10    046050           957    5846#
DH11    046150           964    1196    5857#
DH12    046270           971     990    5871#
DH13    046430           978     984    5888#
DH16    046460           997    5892#
DH17    046560          1004    5903#
DH2     045670           921    1233    5826#
DH20    046650          1011    5913#
DH21    046776          1018    5928#
DH22    047056          1025    5937#
DH23    047155          1032    1128    5948#
DH24    047205          1038    1044    5953#
DH26    047245          1050    1056    5959#
DH3     045760           927     933     939     945    1227    5836#
DH30    047305          1062    5965#
DH31    047365          1068    5974#
DH32    047445          1074    5983#
DH33    047525          1080    5992#
DH34    047565          1086    5998#
DH35    047645          1091    6007#
DH36    047725          1097    1116    1239    6016#
DH37    047765          1103    6022#
DH40    050102          1110    6035#
DH42    050142          1122    6041#
DH44    050202          1134    6047#
DH45    050314          1141    6060#
DH46    050511          1151    1158    1183    6083#
DH50    050566          1165    1190    1203    1221    6092#
DH51    050606          1171    6095#
DH52    050646          1177    6101#
DH57    050764          1209    6115#
DH60    051024          1215    6121#
DH61    051064          1245    1251    1258    1265    6127#
DH62    051122          1271    6132#
DH63    051161          1277    6138#
DH7     046020           951    5842#
DISPLA  001142           782#   1386*   1394*   4415*   4444*
DISPRE  000174           718#   1394
DSWR  = 177570           549#    781    1385
DT1     051220           916    6147#
DT10    051276           959    6156#
DT11    051312           966    1198    6158#
DT12    051326           973     992    6160#
DT13    051344           979     985    6163#
DT16    051364           999    6165#
DT17    051372          1006    6168#
DT2     051234           922    1234    6149#
DT20    051406          1013    6170#
DT21    051430          1020    6173#
DT22    051442          1027    6175#
DT23    051456          1033    1129    6177#
DT24    051466          1039    1045    6179#
DT26    051500          1051    1057    6181#
```

DT3	051254	928	934	940	946	1228	6152#
DT30	051512	1063	6183#				
DT31	051530	1069	6186#				
DT32	051546	1075	6189#				
DT33	051564	1081	6192#				
DT34	051576	1087	6194#				
DT35	051614	1092	6197#				
DT36	051632	1098	1117	6200#			
DT37	051644	1105	6202#				
DT40	051662	1111	6205#				
DT42	051674	1123	6207#				
DT44	051706	1136	6209#				
DT45	051724	1146	6212#				
DT46	051742	1153	1160	1185	6215#		
DT50	051754	1166	1191	1204	1222	6217#	
DT51	051762	1172	6218#				
DT52	051774	1179	6220#				
DT57	052016	1210	6223#				
DT60	052030	1216	6225#				
DT64	052042	1240	6227#				
DT65	052054	1246	1252	1259	1266	6229#	
DT66	052066	1272	6231#				
DT67	052100	1278	6233#				
DT7	051266	952	6154#				
EMTVEC=	000030	638#	1355*	1356*			
EM1	041616	914	5449#				
EM10	042175	956	5494#				
EM11	042241	963	5501#				
EM12	042301	970	5507#				
EM13	042350	977	5514#				
EM14	042405	983	5519#				
EM15	042440	989	5524#				
EM16	042514	996	5532#				
EM17	042556	1003	5538#				
EM2	041656	920	5455#				
EM20	042626	1010	5545#				
EM21	042700	1017	5552#				
EM22	042735	1024	5557#				
EM23	042774	1031	5563#				
EM24	043040	1037	5569#				
EM25	043100	1043	5575#				
EM26	043145	1049	5582#				
EM27	043215	1055	5589#				
EM3	041725	926	5462#				
EM30	043264	1061	5596#				
EM31	043336	1067	1090	1096	1238	5603#	
EM32	043403	1073	5610#				
EM33	043464	1079	5619#				
EM34	043547	1085	5628#				
EM37	043625	1102	5636#				
EM4	041764	932	5468#				
EM40	043672	1109	5643#				
EM41	043741	1115	5650#				
EM42	043774	1121	5655#				
EM43	044033	1127	5661#				
EM44	044100	1133	5668#				

```
EM45    004146              1140    5675#
EM46    004223              1150    5683#
EM47    004266              1157    5689#
EM5     002017               938    5473#
EM50    004331              1164    5695#
EM51    004367              1170    5701#
EM52    004445              1176    5709#
EM53    004523              1182    5717#
EM54    004566              1189    5723#
EM55    004624              1195    5728#
EM56    004662              1202    5733#
EM57    004716              1208    5738#
EM6     002072               944    5481#
EM60    004773              1214    5746#
EM61    005036              1220    5752#
EM62    005114              1226    5760#
EM63    005165              1232    5767#
EM64    005225              1244    5773#
EM65    005266              1250    5779#
EM66    005351              1257    5788#
EM67    005441              1264    5798#
EM7     002137               950    5488#
EM70    005506              1270    5805#
EM71    005562              1276    5813#
ERRTYP  035556              4454    4500#
ERRVEC= 000004               631#   1383    1384#   1395#   1426#   1427#   2890#   2900#   4376    4377#   4379#   4382#
FORMPA  036354              2353    2379    2405    2432    2455    2497    2523    2549    2575    2601    2672    2713    4725#
GMS   = ******  U             717    1421    4309    4316    5381    5382    5383    5384    5385    5386    5388    5390    5391
                            5392    5393    5394
GTSWR = 104407              1416    4949    5388#
HT    = 000011               541#   5000    5041
IOTVEC= 000020               636#   1353#   1354#
KERSTK= 001100               704#   1537    1541    1543    2075    2084    2093    2102    2216    2892    3070    3142    3146
                            3159    3217    3221    3234    3302    3364    3369    3431    3460    3519    3538    3598    3659
                            3753    4085    4217
KIPAR0= 172340               691#   1618    1877    1971    2083    2136    2324    3717#   4626    4664    4725
KIPAR1= 172342               692#   3718#
KIPAR2= 172344               693#   3719#
KIPAR3= 172346               694#   2251#   2748    2928#   3289#   3720#
KIPAR4= 172350               695#   2252#   2342#   2368#   2394#   2421#   2445#   2486#   2512#   2538#   2564#   2590#   2642#
                           2692#   2699#   2929#   3290#   3587#   3721#   3741#   3893#   4071#
KIPAR5= 172352               696#   2643#   2693#   2700#   3339#
KIPAR6= 172354               697#
KIPAR7= 172356               698#   2253#   3722#
KIPDR0= 172300               680#   1658    1924    2019    2074    2125    2197    2205#   2331    2752    2756    2770    3726
                            4622    4650
KIPDR1= 172302               681#
KIPDR2= 172304               682#
KIPDR3= 172306               683#   2254#   2272#   2889#   2893    2932#   3135#   3291#
KIPDR4= 172310               684#   2255#   2690#   2697#   2939#   3006#   3140#   3215#   3292#   3317#   3340#   3497#   3589#
                           3619#   3740#   3746#   3891#   3917#   3919#   3934#   3936#   3949#   3951#   3965#   3967#   3983#
                           3985#   4000#   4002#   4019#   4021#   4070#
KIPDR5= 172312               685#   2691#   2698#   3136#   3341#   3420#   3498#
KIPDR6= 172314               686#
KIPDR7= 172316               687#   2256#   2878#   2881
LF    = 000012               542#   5035    5041
```

```
LOOP    020466          1424#  4351
MGMERR  002400          1319#  1428    2218    2271    2703    2986    3047    3089    3195    3270    3318    3499    3618
                        3692   3859    4030    4186
MGMFLG  002402          1320#  1334#   1432#
MMVEC = 000250           647#  1428#   1429#   2201#   2202#   2218#   2219#   2257#   2271#   2662#   2703#   2938#   2986#
                        3005#  3047#   3062#   3089#   3139#   3152#   3195#   3214#   3227#   3270#   3295#   3318#   3361#
                        3367#  3419#   3451#   3499#   3745#   3859#   3895#   4030#   4078#   4186#
ORADR   001376           891#  1581#   1600#   1621#   1640#   1661#   1680#   1701#   1720#   1741#   1760#   6156
PBAHI   001410           896#  4744#
PBALO   001406           895#  4735#   4736#   4741#   6168    6170    6220
PIRQ  = 177772           548#
PIRQVE= 000240           642#
PR0   = 000000           565#
PR1   = 000040           566#
PR2   = 000100           567#
PR3   = 000140           568#
PR4   = 000200           569#
PR5   = 000240           570#
PR6   = 000300           571#
PR7   = 000340           572#
PS    = 177776           545#   546
PSW   = 177776           546#  1472#   1473#   1474    1478#   1486#   1498#   1500#   1501    1506#   1512#   1514#   1515
                        1519#  1536#   1538#   1540#   1836#   1846#   2417    2441    2639#   2663    2684    2694#   2696#
                        2812#  2860#   2943    2976#   2982#   2984#   3010    3043#   3045#   3063#   3518#   3523#   3530
                        3532#  3537#   3540#   3542#   3558#   3562    3564#   3649#   3658#   3660#   3662#   3750#   3771#
                        3783#  3795#   3806#   3820#   3834#   3847#   3898#   3909#   3930#   3946#   3961#   3978#   3996#
                        4013#  4069#   4080#   4099#   4112#   4124#   4135#   4148#   4162#   4175#   4187#   4214#   4257#
                        4584    4586    4726
PWRMSG  041554          5430    5439#
PWRVEC= 000024            637#  1359#   1360#   5399#   5400#   5409#   5415#   5427#   5428#
RDCHR = 104411          4869    5391#
RDLIN = 104412          5392#
RESREG= 104414          5337    5394#
RESVEC= 000010            632#  1370#   1377#
SAVREG= 104413          5322    5393#
SETREG  036070          2076    2085    2094    2103    2123    4621#
SR0   = 177572            651#  1330    1332#   1434#   1578    1780    1837#   1839    1845#   2207#   2209    2211#   2217#
                        2260#  2278#   2279    2284#   2344#   2370#   2396#   2423#   2446#   2488#   2514#   2540#   2566#
                        2592#  2660#   2714    2716#   2879    2880#   2957    2972#   3024    3039#   3072    3088#   3160
                        3176#  3183    3185#   3235    3251#   3258    3260#   3310#   3365    3370    3386    3400#   3425
                        3432    3435    3445    3454    3467    3475    3479    3486#   3600    3616#   3655    3657#   3866
                        3868#  4036    4038#   4194    4196#
SR1   = 177574            652#  1858
SR2   = 177576            653#  1331    1790    1811    2280    2715    2958    3025    3081    3161    3184    3236    3259
                        3303    3343    3352    3366    3371    3390    3401    3426    3436    3441    3446    3455    3461
                        3487    3601    3656    3867    4037    4195
SR3   = 172516            654#
STACK = 001100            536#   704     705    1351    1425    4258    4277
START   020000            721   1344#   5432
STKLMT= 177774            547#
SWR     001140            781#  1349    1385#   1387    1393#   1400#   1414    4340    4371    4385    4387    4393    4400
                        4445    4452    4464    4468    4757    4802#   5407    5420#
SWREG   000176            719#  1393    1414    4757    4770
SW0   = 000001            600#
SW00  = 000001            590#   600
SW01  = 000002            589#   599
```

```
SW02  = 000004          588#    598
SW03  = 000010          587#    597
SW04  = 000020          586#    596
SW05  = 000040          585#    595
SW06  = 000100          584#    594
SW07  = 000200          583#    593
SW08  = 000400          582#    592
SW09  = 001000          581#    591
SW1   = 000002          599#
SW10  = 002000          580#
SW11  = 004000          579#
SW12  = 010000          578#
SW13  = 020000          577#
SW14  = 040000          576#
SW15  = 100000          575#
SW2   = 000004          598#
SW3   = 000010          597#
SW4   = 000020          596#
SW5   = 000040          595#
SW6   = 000100          594#
SW7   = 000200          593#
SW8   = 000400          592#
SW9   = 001000          591#
TBIT  = 000020          702#    4584    4589    4602
TBITPS  001372          889#    1433*   4587*   4602    4604    4605*
TBITVE= 000014          633#    1367*   1368*
TESTNO  001352          881#    4441*   6147    6149    6152    6154    6156    6158    6160    6163    6165    6168    6170
                       6173    6175    6177    6179    6181    6183    6186    6189    6192    6194    6197    6200    6202
                       6205    6207    6209    6212    6215    6217    6218    6220    6223    6225    6227    6229    6231
                       6233
TIMERR  002332         1292#   1426    1591    1631    1671    1711    1751    2900    3543    3617    3690
TIMFLG  002334         1293#   1304*   1431*
TKVEC = 000060          640#
TOFF    036002         2196    2249    2746    2813    2875    3516    3639    4584#
TON     036036         2221    2274    2792    2859    2902    3545    3695    4602#
TONUM   001400          892#   1582*   1588    1603*   1622*   1628    1643*   1662*   1668    1683*   1702*   1708    1723*
                       1742*   1748    1763*   6156
TPVEC = 000064          641#
TRAPPC  001356          883#   1300*   1306    1327*   1336    2282*   2290    2707*   2723    3181*   3191    3256*   3266
                       3864*   3874    4034*   4044    4192*   4202    6147    6149
TRAPPS  001360          884#   1301*   1305    1328*   1335    2283*   2289    2708*   2722    3182*   3190    3257*   3265
                       3865*   3873    4035*   4043    4193*   4201    6147    6149
TRAPVE= 000034          639#   1357*   1358*
TRTVEC= 000014          634#
TST1    020550         1445#
TST10   021514         1672    1694#
TST11   021646         1712    1734#
TST12   022000         1752    1778#
TST13   022142         1834#
TST14   022200         1856#
TST15   022216         1859    1875#
TST16   022340         1922#
TST17   022462         1969#
TST2    020622         1469#
TST20   022612         2017#
TST21   022742         2070#
```

```
TST22   023162          2120#
TST23   023332          2195#
TST24   023466          2248#
TST25   023724          2275    2322#
TST26   024454          2477#
TST27   025140          2637#
TST3    020706          1496#
TST30   025604          2705    2744#
TST31   026000          2811#
TST32   026206          2873#
TST33   026336          2925#
TST34   026632          2998#
TST35   027042          3060#
TST36   027166          3134#
TST37   027470          3211#
TST4    021030          1535#
TST40   027756          3288#
TST41   030144          3338#
TST42   031122          3515#
TST43   031310          3554#
TST44   031372          3586#
TST45   031570          3638#
TST46   032112          3716#
TST47   032760          3861    3890#
TST5    021076          1542    1574#
TST50   033640          4031    4061#
TST51   034404          4189    4213#
TST52   034510          4256#
TST6    021230          1592    1614#
TST7    021362          1632    1654#
TYPBN = 104406          4549    5386#
TYPDS = 104405          4313    4320    4542    4948    5385#
TYPE  = 104401          1408    4307    4314    4321    4447    4455    4501    4518    4520    4523    4525    4557    4562
                        4568    4768    4769    4772    4783    4793    4804    4823    4873    4882    4888    4893    4897
                        4902    4903    4905    4908    4912    4942    4946    5005    5114    5182    5257    5381#   5429
TYPOC = 104402          4509    4535    4771    4945    5382#
TYPON = 104404          5384#
TYPOS = 104403          5383#
UIPAR0= 177640          669#    1698    1907    2001    2101    2159    2644    3520*   3539*   3733*   4634    4692    4728
UIPAR1= 177642          670#    3734*
UIPAR2= 177644          671#    3735*
UIPAR3= 177646          672#    2815    2930*   3640*   3736*
UIPAR4= 177650          673#    2658*   268( *  2701*   2931*   3641*   3742*   3894*   4072*
UIPAR5= 177652          674#    2659*   2687*   2702*
UIPAR6= 177654          675#
UIPAR7= 177656          676#    1905    1999    3737*
UIPDR0= 177600          658#    1738    1954    2049    2092    2147    2651    2819    2823    2837    3730    4065    4630
                        4678
UIPDR1= 177602          659#
UIPDR2= 177604          660#
UIPDR3= 177606          661#    2933*   3642*   3693*
UIPDR4= 177610          662#    2656*   2688*   2940*   3007*   3643*   3892*    079*
UIPDR5= 177612          663#    2657*   2689*
UIPDR6= 177614          664#
UIPDR7= 177616          665#    1952    2047
USESTK= 000700          705#    1539    3524    3541    3661    3756    3910    4082    4220    4243
```

```
VIRT1   001402          893# 2352# 2378# 2404# 2431# 2454# 2496# 2522# 2548# 2574# 2600# 2671# 2679
                       2712# 4729  4735  6168  6170  6220
VIRT2   001404          894# 2670# 6170
WASR6   001354          882# 1302# 1329# 2281# 6147  6149
WASSR0  001366          887# 1330# 2279# 2714# 2957# 2959  3024# 3026  3072# 3074  3160# 3163  3183#
                       3235# 3238  3258# 3370# 3372  3386# 3387  3425# 3435# 3445# 3454# 3467# 3479#
                       3600# 3603  3655# 3674  3866# 4036# 4194# 6149  6183  6189  6194  6202  6205
                       6209  6212  6218  6220  6223  6229  6233
WASSR2  001370          888# 1331# 1790# 1792  1811# 1812  2280# 2715# 2958# 2966  3025# 3033  3081#
                       3082  3161# 3170  3184# 3236# 3245  3259# 3303# 3304  3343# 3345  3352# 3354
                       3371# 3375  3390# 3391  3401# 3403  3426# 3436# 3446# 3455# 3461# 3462  3487#
                       3488  3601# 3607  3656# 3677  3867# 4037# 4195# 6149  6186  6189  6197  6200
                       6202  6205  6209  6212  6218  6220  6227  6229  6231
WBIT  = 000100          703# 2762  2771  2782  2829  2838  2849
$APTHD  000204          741   747#
$ASTAT= ******  U       5075  5090
$ATYC   040116          5046  5048#
$ATY1   040072          5044#
$ATY3   040100          4990  5045#
$ATY4   040110          4460  5047#
$AUTOB  001134          778#  1418#  4765  4929
$BASE   001302          856#
$BDADR  001122          773#
$BDBRT  001126          775#
$BELL   001216          807#  4447  4480
$BIN    040412          5110# 5113# 5114  5121#
$CDW1   001306          858#
$CDW2   001310          859#
$CHARC  040066          5007# 5017# 5024  5033# 5038#
$CKSUM  036466          4757# 5390
$CLR.T  034750          4326  4329#
$CMTRG  001100          761#  1346  1347  1355  1361  1362  1363
$CM1  = 000006          793#   794#  795#  796#  797#  798#  799#
$CM2  = 000014          793#   794#  795#  796#  797#  798#  799#
$CM3  = 000006          791#   793
$CMM  = 000006          799#   800#  801#  802#  803#  804#  805#
$CNTLC  037416          4783  4873  4922#
$CNTLG  037430          4768  4924#
$CNTLU  037423          4793  4897  4923#
$CPUOP  001254          830#
$CRLF   001223          809#  4321  4455  4480  4501  4520  4525  4568  4804  4902  4922  5006  5041
$DBLK   041056          5223  5257  5265#
$DB20   041162          4554  5322#
$DDW0   001312          860#
$DDW1   001314          861#
$DDW10  001336          870#
$DDW11  001340          871#
$DDW12  001342          872#
$DDW13  001344          873#
$DDW14  001346          874#
$DDW15  001360          875#
$DDW2   001316          862#
$DDW3   001320          863#
$DDW4   001322          864#
$DDW5   001324          865#
$DDW6   001326          866#
```

```
$DDN7   001330          867#
$DDN8   001332          868#
$DDN9   001334          869#
$DEVCT  001236          821#
$DEVM   001304          857#
$DORGN  034770         4303    4324    4331    4337#
$DTBL   041046         5226    5261#
$ENDRD  034760          728    1406    4333#   4475
$ENDCT  034606         1361    4305#
$ENULL  035034         4353#
$ENV    001246          826#   1412    4457    4985    5053    5077
$ENVM   001247          827#   1398    4987    4992    5055
$EOP    034552         4295#   4950
$EOPCT  034600         1361*   1302#   4306
$ERFLG  001103          764#    360    4389    4391    4397*   4419    4442*   4480
$ERMAX  001115          770#    364*   4391    4414*   4419
$ERROR  035320         1355    4433#
$ERRPC  001116          771#   4449*   4450*   4451    4480    4507    6147    6149    6152    6154    6156    6158    6160
                       6163    6165    6168    6170    6173    6175    6177    6179    6181    6183    6186    6189    6192
                       6194    6197    6200    6202    6205    6207    6209    6212    6215    6217    6218    6220    6223
                       6225    6227    6229    6231    6233
$ERRTB  001412          911#   4515
$ERTTL  001112          768#   4318    4322*   4448*   4480
$ESCAP  001214          806#   1363*   4413*   4471    4473    4480
$ETABL  001246          825#
$ETEND  001352          753     878#
$FATAL  001230          818#   5081*
$FFLG   040336         5044*   5047*   5075    5084*   5092#
$FILLC  001156          789#   5010    5041
$FILLS  001155          788#   5041
$GOADR  001120          772#
$GOORT  001124          774#
$GET42  034732         4323#
$GTSWR  036536         4769#   5388
$HD     =  000000       521
$HIBTS  000204          748#
$ICNT   001104          765#   4404*   4405    4407*   4418
$ILLUP  041546         5399    5415    5436#
$INTRG  001135          779#   4785    4805    4929
$ITEMB  001114          769#   4451*   4459    4480    4504
$LF     001224          810#   4480    4912    4922    5041
$LFLG   040335         5085*   5091#
$LOOP   035026         4345    4349#
$LPADR  001106          766#   1379*   4395*   4411*   4416    4418
$LPERR  001110          767#   1380*   1446*   1461*   1470*   1485*   1497*   1511*   1524*   1576*   1587*   1616*   1627*
                       1656*   1667*   1696*   1707*   1736*   1747*   1789*   1798*   1822*   1879*   1888*   1909*   1926*
                       1935*   1956*   1972*   1985*   2003*   2020*   2033*   2051*   2072*   2081*   2090*   2099*   2108*
                       2206*   2220*   2250*   2273*   2337*   2363*   2389*   2416*   2440*   2463*   2481*   2507*   2533*
                       2559*   2585*   2611*   2661*   2704*   2755*   2791*   2822*   2858*   2877*   2888*   2901*   2941*
                       2985*   3008*   3046*   3061*   3090*   3141*   3145*   3151*   3194*   3216*   3220*   3226*   3269*
                       3296*   3316*   3342*   3351*   3360*   3421*   3496*   3517*   3544*   3556*   3571*   3592*   3620*
                       3648*   3694*   3738*   3749*   3769*   3782*   3794*   3805*   3819*   3833*   3846*   3860*   3913*
                       3929*   3945*   3960*   3977*   3995*   4012*   4029*   4068*   4098*   4111*   4123*   4134*   4147*
                       4161*   4174*   4188*   4232*   4245*   4395    4412*   4418    4470
$MADR1  001260          843#
$MADR2  001264          847#
```

```
SMADR3  001270            850#
SMADR4  001274            853#
SMAIL   001226            749      753      816#     1397     1412     4410     4457     4985
SMAMS1  001256            837#
SMAMS2  001262            845#
SMAMS3  001266            848#
SMAMS4  001272            851#
SMBADR  000206            749#
SMFLG   040334           5045#     5051     5086#    5090#
SMNEW   037446           4[illegible]2     4927#
SMSGAD  001242             23#     5061#    5064
SMSGLG  001244            824#     5066#
SMSGTY  001226            817#     5059     5067#    5079     5083#
SMSWR   037435           4769      4925#
SMTYP1  001257            838#
SMTYP2  001263            846#
SMTYP3  001267            849#
SMTYP4  001273            852#
SMXCNT  035316           4408      4418#
SNULL   001154             787#    5012     5041
SNWTST= 000001           1438#     1440     1463#    1465     1488#    1490     1526#    1528     1564#    1566     1606#    1608     1646#
                         1648      1686#    1688     1726#    1728     1766#    1768     1824#    1826     1848#    1850     1865#    1867
                         1911#     1913     1960#    1962     2005#    2007     2053#    2055     2110#    2112     2182#    2184     2235#
                         2237      2305#    2307     2465#    2467     2613#    2615     2727#    2729     2794#    2796     2862#    2864
                         2916#     2918     2989#    2991     3049#    3051     3122#    3124     3199#    3201     3274#    3276     3324#
                         3326      3503#    3505     3547#    3549     3574#    3576     3623#    3625     3704#    3706     3878#    3880
                         4049#     4051     4205#    4207     4247#    4249
SOCNT   040636           5154#     5183#    5196#
SOCTVL  041264           5324      5349#
SOMODE  040640           5149#     5153#    5158     5161#    5172#    5198#
SOVER   035302           4372      4388     4396     4406     4415#
SPASS   001234            820#     1397#    4299#    4300#    4311     4352     4402     4419     4947
SPASTN  000212            751#
SPMRAD  041530           5432#
SPMRON  041370           1359      5399#    5427
SPMRMB  041524           5430#
SPMRUP  041442           5409      5415#
SQUES   001222            808#     4480     4823     4905     4922     5041
SRDCNR  037010           4836#     539!
SRDDEC= ******  U        5393
SRDLIN  037130           4864#     5392
SRDOCT= ******  U        5393
SRDSZ = 000010           4857#
SREGAD  001160            791#
SREG0   001162            793#     4435#    6152     6154     6158     6160     6163     6165     6175     6189     6192     6194     6197
                         6209      6215
SREG1   001164            794#     4436#    6152     6158     6160     6165     6179     6200     6207     6209     6212     6215     6223
                         6225
SREG2   001166            795#     4437#    6160     6165     6179     6181     6183     6186     6194     6207     6225
SREG3   001170            796#     4438#    6173     6175     6183     6197     6212
SREG4   001172            797#     4439#    6168     6170     6186     6189     6192     6194     6197     6220     6227
SREG5   001174            798#     4440#    6165     6170     6173     6175     6177
SRESRE  041124           5299#     5394
SRTNAD  035030           4351#
SRTRN   035024           1367      1369#    1374#    4327     4346#
SR2A  = ******  U        5395
```

```
$SAVRE  041066          5283#  5393
$SAVR6  041552          5408#  5416   5417#  5418#  5438#
$SCOPE  035040          1353   4369#
$SETUP= 000137           711#  1352   1353   1355   1357   1359   1361   1362   1363   1365   1379   1406   1409
                        4297   4370   4434   4467   4475   4752   4929   5434
$STUP  = 177777          711#
$SVLAD  035246          4380   4409#
$SVPC  = 000204          726#   731
$SWR   = 177400          515#   521    525    526    527    528    529    530    531    532    805    806    807
                        1362   1363   1365   1379   1380   1446   1470   1497   1536   1575   1615   1655   1695
                        1735   1779   1835   1857   1876   1923   1970   2018   2071   2121   2196   2249   2323
                        2478   2638   2745   2812   2874   2926   2999   3061   3135   3212   3289   3339   3516
                        3555   3587   3639   3717   3891   4062   4214   4257   4291   4298   4325   4338   4352
                        4361   4362   4363   4364   4365   4371   4383   4385   4386   4389   4390   4391   4398
                        4399   4400   4412   4415   4418   4425   4426   4427   4428   4429   4445   4452   4464
                        4468   4480   5433
$SWREG  001250           828#  1400
$SWRMK= 000000           532    533   4365   4366   4387
$TBIT   035032          1378#  4342#  4352#  5434#
$TESTN  001232           819#  4410#
$TIMES  001212           805#  1362#  4298#  4398#  4405   4408#  4418
$TKB    001146           784#  4750   4761   4778   4840   4846
$TKS    001144           783#  4750   4759   4775   4787#  4807#  4838   4844
$TMP0   001176           799#  2343#  2348   2369#  2374   2395#  2400   2487#  2492   2513#  2518   2539#  2544
                        2565#  2570   2591#  2596   2663#  2943#  2973   3010#  3040   3365#  3372   3663#  3669#
                        3673#  3676#  3680#  3681   6170   6181   6183   6186   6202   6220
$TMP1   001200           800#  3362#  3374#  3377#  3378   3384#  3389#  3393#  3394   3422#  3424#  3430#  3434#
                        3440#  3444#  3450#  3453#  3459#  3466#  3473#  3478#  3485#  3491#  3599#  3605#  3609#
                        3610
$TMP2   001202           801#  3366#  3375   3821#  3822   3848#  3849   3980   3981#  4015#  4017   4149#  4150
                        4176#  4177   6202
$TMP3   001204           802#
$TMP4   001206           803#
$TMP5   001210           804#  4940#  4941#  4947
$TN    = 000053          515#   521   1438   1446#  1463   1470#  1488   1497#  1526   1536#  1542   1564   1575#
                        1592   1606   1615#  1632   1646   1655#  1672   1686   1695#  1712   1726   1735#  1752
                        1766   1779#  1824   1835#  1848   1857#  1859   1865   1876#  1911   1923#  1960   1970#
                        2005   2018#  2053   2071#  2110   2121#  2182   2196#  2235   2249#  2275   2305   2323#
                        2465   2478#  2613   2638#  2705   2727   2745#  2794   2812#  2862   2874#  2916   2926#
                        2989   2999#  3049   3061#  3122   3135#  3199   3212#  3274   3289#  3324   3339#  3503
                        3516#  3547   3555#  3574   3587#  3623   3639#  3704   3717#  3861   3878   3891#  4031
                        4049   4062#  4189   4205   4214#  4247   4257#
$TPB    001152           786#  5030#  5041
$TPFLG  001157           790#  4979   5041
$TPS    001150           785#  5028   5041
$TRAP   041302          1357   5358#
$TRAP2  041324          5369#  5380
$TRP   = 000015          5373#  5382#  5383#  5384#  5385#  5386#  5387#  5388   5389#  5390   5391#  5392#  5393#
                        5394#  5395#
$TRPAD  041336          5363   5380#
$TSTM   000210           750#
$TSTNM  001102           763#  4297#  4360   4387   4409#  4410   4415   4419   4441   4444   4480   4943
$TTYIN  037406          4866   4867   4885   4903   4917   4921#
$TYPBN  040340          5107#  5386
$TYPOS  040642          5211#  5385
$TYPE   037610          4979#  5072   5373   5391
```

```
$TYPEC  000022          4809    5009    5016    5023    5028#   5029
$TYPEX  000070          5034    5036    5039#
$TYPOC  000440          5152#   5382
$TYPON  000454          5151    5154#   5384
$TYPOS  000414          5147#   5383
$UNIT   001240           822#
$UNITM  000214           752#
$USWR   001252           829#
$VECT1  001276           854#
$VECT2  001300           855#
$XTSTR  035052          4374#
$$GET4= 000001          4325#   4330
$OFILL  000637          5148*   5152*   5162    5197#
$40CAT= ******  U       4371    4454
      = 052267           713#    717#    726     727#    729#    731#    737     738#    740#    742#    760#    811     1342#
                        1350    1379    1380    4310#   4352    4354#   4418    4419    4480    4571#   4750    4921#   4922
                        4929#   5041    5093#   5265#   5349#   5411    5437    5445#
 $ASTA= ******  U       5045    5048
 $X   = 000204           737#    742
```

ADDTST 710# 2337 2363 2389 2481 2507 2533 2559 2585
COMMEM 1# 643#
ENDCOM 1# 643#
ERROR 537# 1303 1333 1454 1479 1507 1520 1545 1590 1595 1630 1635 1670 1675 1710
1715 1750 1755 1785 1794 1806 1815 1841 1860 1883 1896 1930 1943 1981 1993
2029 2041 2130 2141 2152 2164 2212 2266 2285 2354 2380 2406 2433 2456 2498
2524 2550 2576 2602 2673 2717 2764 2775 2784 2831 2842 2851 2884 2896 2945
2953 2961 2968 3012 3020 3028 3035 3065 3076 3084 3154 3165 3172 3186 3229
3240 3247 3261 3306 3347 3356 3380 3396 3405 3427 3437 3447 3456 3469 3481
3493 3533 3567 3594 3612 3683 3759 3764 3777 3789 3800 3812 3827 3840 3855
3869 3905 3923 3940 3955 3971 3989 4006 4025 4039 4088 4093 4106 4118 4129
4141 4155 4168 4182 4197 4223 4228 4239 4266 4273 4658 4672 4686 4700
ESCAPE 1# 643#
GETPRI 1# 643# 4338
GETSWR 1# 643# 1409#
MSG1 1438# 1440
MSG10 1686# 1688
MSG11 1726# 1728
MSG12 1766# 1768
MSG13 1824# 1826
MSG14 1848# 1850
MSG15 1865# 1867
MSG16 1911# 1913
MSG17 1960# 1962
MSG2 1463# 1465
MSG20 2005# 2007
MSG21 2053# 2055
MSG21A 2110# 2112
MSG21B 2182# 2184
MSG21C 2236# 2237
MSG22 2306# 2307
MSG23 2466# 2467
MSG24 2613# 2615
MSG25 2727# 2729
MSG26 2794# 2796
MSG27 2862# 2864
MSG3 1488# 1490
MSG30 2916# 2918
MSG31 2989# 2991
MSG31A 3049# 3051
MSG32 3121# 3124
MSG33 3198# 3201
MSG34 3273# 3276
MSG35 3321# 3326
MSG36 3502# 3505
MSG36A 3547# 3549
MSG37 3573# 3576
MSG4 1526# 1528
MSG40 3622# 3625
MSG41 3704# 3706
MSG42 3878# 3880
MSG43 4049# 4051
MSG44 4206# 4207
MSG45 4247# 4249
MSG5 1563# 1566
MSG6 1606# 1608

```
MSG7    16468#  1648#
MULT      1#    6438#
NEWTST    1#    6438#  1438   1463   1488   1526   1564   1606   1646   1686   1726   1766   1824   1848   1865
        1911    1960   2005   2053   2110   2182   2235   2305   2465   2613   2727   2794   2862   2916   2989
        3049    3122   3199   3274   3324   3503   3547   3574   3623   3704   3878   4049   4205   4247
POP       1#    6438#  5087   5088   5252   5304   5420   5421
PUSH      1#    6438#  5048   5050   5071   5211   5284   5401   5407
REPORT    1#    6438#
SAVR    7108#   4435#
SCOPE   5388#   1445   1469   1496   1535   1574   1614   1654   1694   1734   1778   1834   1856   1875   1922
        1969    2017   2070   2120   2195   2248   2322   2477   2637   2744   2811   2873   2925   2998   3060
        3134    3211   3288   3338   3515   3554   3586   3638   3716   3890   4061   4213   4256   4296
SETPRI    1#    6438#
SETTRA  5373#   5382   5383   5384   5385   5386   5388   5390   5391   5392   5393   5394
SETUP     1#    6438#  1344
SKIP      1#    6438#  1542   1592   1632   1672   1712   1752   1859   2275   2705   3861   4031   4189
SLASH     1#    6438#
SPACE   6438#
STARS     1#    6438#   724    734    736    743    756    811    814   1284   1291   1311   1318   1438   1444
        1463    1468   1488   1495   1526   1534   1550   1561   1564   1573   1606   1613   1646   1653   1686
        1693    1726   1733   1766   1777   1824   1833   1848   1855   1865   1874   1911   1921   1960   1968
        2005    2016   2053   2069   2110   2119   2182   2194   2235   2247   2305   2321   2465   2476   2613
        2636    2727   2743   2794   2810   2862   2872   2904   2914   2916   2924   2989   2997   3049   3059
        3103    3118   3122   3133   3199   3210   3274   3287   3324   3337   3503   3514   3547   3553   3574
        3585    3623   3637   3696   3701   3704   3715   3878   3889   4049   4060   4205   4212   4247   4255
        4287    4357   4421   4482   4576   4583   4595   4601   4611   461,   4640   4648   4709   4723   4749
        4752    4828   4857   4964   5043   5100   5124   5201   5268   5313   5352   5397   5413
SWRSU     1#    6438#  13818#
TIMMSG  7108#   1608#  1648   1688   1728
TIMTST  7108#   1615   1655   1695   1735
TRMTRP  5373#
TYPBIN    1#    6438#  4548
TYPDEC    1#    6438#  4311   4318   4541   4947
TYPNRM    1#    6438#  1402
TYPNUM    1#    6438#
TYPOCS    1#    6438#
TYPOCT    1#    6438#  4507   4534   4770   4944
TYPTXT    1#    6438#  4307   4314
USER    7108#   880#
WMSG    7108#  2729   2796
WTST    7108#  2745   2813
$$CMRE  7548#   793    794    795    796    797    798
$$CMTM  7648#   799    800    801    802    803    804
$$ESCA    1#    6438#
$$NEWT    1#    6438#  1438   1463   1488   1526   1564   1606   1646   1686   1726   1766   1824   1848   1865
        1911    1960   2005   2053   2110   2182   2235   2305   2465   2613   2727   2794   2862   2916   2989
        3049    3122   3199   3274   3324   3503   3547   3574   3623   3704   3878   4049   4205   4247
$$SET   5373#   5382   5383   5384   5385   5386   5388   5390   5391   5392   5393   5394
$$SETM  13978#
$$SETU  13668#
$$SKIP    1#    6438#  1542   1592   1632   1672   1712   1752   1859   2275   2705   3861   4031   4189
. EQUAT   1#    5158#   533
  HEADE   1#    5158#   516
. KT11    1#    5158#   643
  SETUP   1#    5158#   711
  SWRHI   1#    5158#   521
```

```
 $ABLO    5338
 $ACT1      18     5158     722
 $APTB      18     8128
 $APTH      18     5158     732
 $APTY      18     5158    5041
 $ASTA      18
 $CATC      18     5158     711
 $CMTA      18     5158     754
 $DB2D      18
 $DB20      18     5158    5311
 $DIV       18
 $EOP       18     5158    4285
 $ERRO      18     5158    4419
 $ERRT      18     5158
 $MULT      18
 $POWE      18     5158    5395
 $RAND      18
 $%DDE      18
 $RDOC      18
 $?ERD      18     5158    4747
 $RZAZ      18
 $SAVE      18     5158    5266
 $SB2D      18
 $SB20      18
 $SCOP      18     5158    4355
 $SIZE      18
 $SUPR      18
 $TRAP      18     5158    5350
 $TYPB      18     5158    5098
 $TYPD      18     5158    5199
 $TYPE      18     5158    4962
 $TYPO      18     5158    5122
 $4DCA      18
 1170       18

  ABS   052267      000

ERRORS DETECTED   0

CFKTHB,CFKTHB LST/CRF/SOL=CFKTHB SML,CFKTHB P11
RUN-TIME   22 30 2 SECONDS
RUN-TIME RATIO  266/55=4 8
CORE USED   39K  (77 PAGES)
```