PDP11/23

F-11 MMU DIAGNOSTIC

CJKDAA0

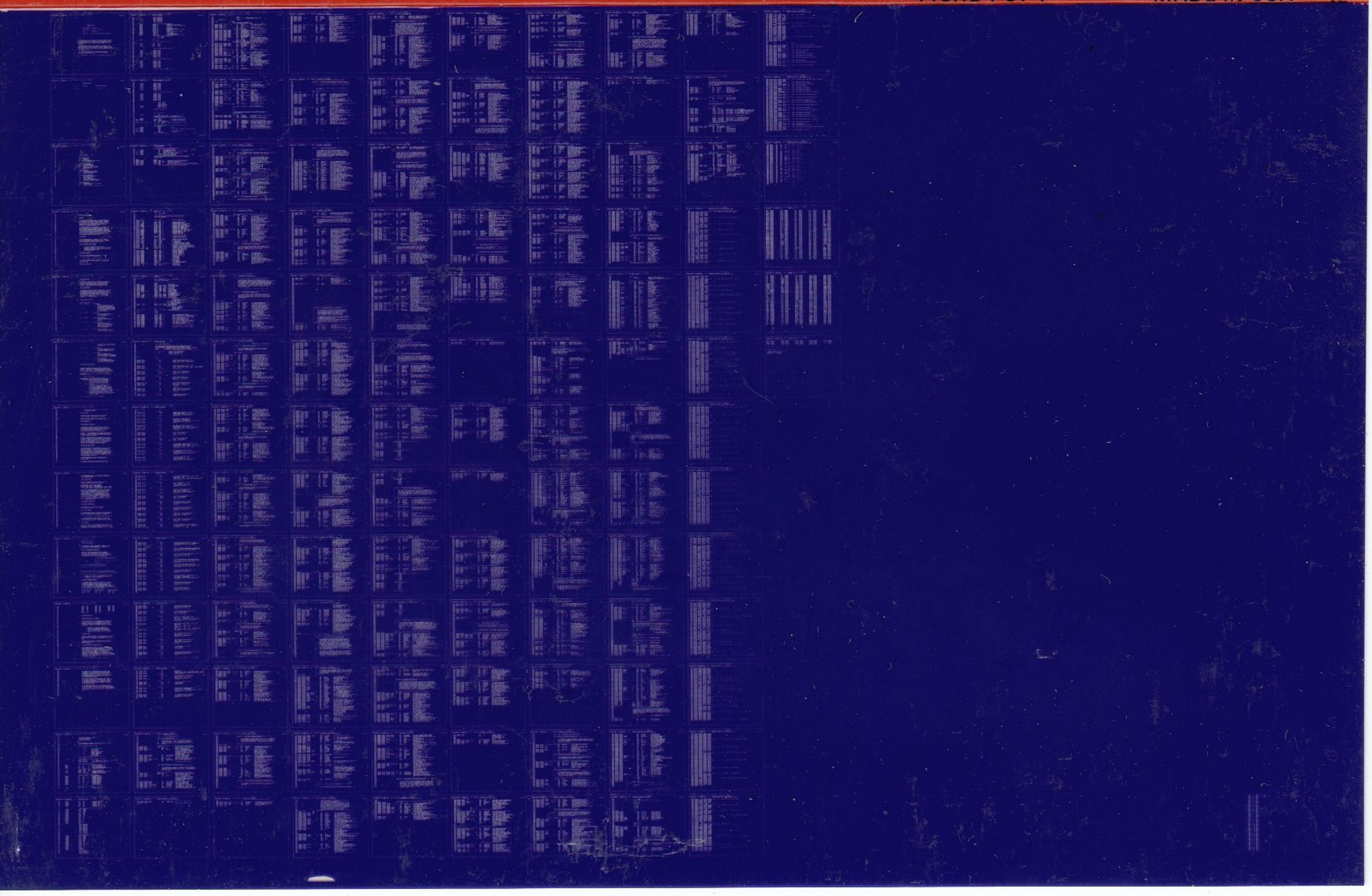
AH-F137A-MC

FICHE 1 OF 1

MAR 1979

digital

MADE IN USA



```
                                .REM    a
```

IDENTIFICATION

PRODUCT CODE: AC-F138A-MC

PRODUCT NAME: CJKDAA0 F-11 MMU DIAG

DATE: 12-JAN-1979

MAINTAINER: SMALL SYSTEMS DIAGNOSTIC ENGINEERING

```
                                PROGRAM HISTORY
                                ---------------

                    DATE        REVISION        REASON FOR REVISION
                    ----        --------        -------------------

                    12-JAN-79       A           FIRST RELEASE
```

TABLE OF CONTENTS

1.0 PROGRAM INFORMATION

1.1 ABSTRACT

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

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

1.2 REQUIREMENTS

A KDF11-A 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 DIAGNOSTIC SHOULD BE RUN:

MD-11-CJKDB   FONZ-11 CPU 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. SINCE 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)

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
                                        ---------------------------


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

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

                                                SWR = XXXXXX  NEW =

                                        WHERE "XXXXXX" IS THE CURRENT OCTAL CONTENTS OF LOC. 176.
                                        THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING:
                                            XXXXXX<CR>  ONE TO SIX OCTAL DIGITS FOLLOWED BY A
                                                        CARRIAGE RETURN WHICH WILL BE LOADED
                                                        AS THE NEW VALUE FOR THE SWITCH REG.
                                                <CR>    JUST A <CR>, LEAVES THE SWITCH REG.
                                                        AS IT IS.
                                                XXX^U   A CONTROL-U (^U) WILL CAUSE ALL OF THE
                                                        DIGITS TYPED SO FAR TO BE IGNORED.
                                                ^C      WILL CAUSE THE PROGRAM TO TYPE THE PRESENT
                                                        TEST AND PASS NUMBERS, REQUEST A NEW VALUE
                                                        FOR THE SWITCH REG., AND JUMP TO THE END-
                                                        OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY
                                                        TO THE NEXT PASS WITH A NEW SW. REG. VALUE
                                            <ILL.CHAR>  ANY CHARACTER TYPED WHICH IS NOT ANY OF THE
                                                        ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM
                                                        TO TYPE A "?<CRLF>" AND REACT AS THOUGH A
                                                         U HAD BEEN TYPED.

                                        NOTE:   RECOGNITION OF A ^G MAY BE HAMPERED BY
```

```
                                        -----   EXECUTION OF A COUPLE "RESET" INSTRUCTIONS
                                                WITHIN THE PROGRAM.

                                2.5     EXECUTION TIMES
                                        ---------------

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

                                        THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS
                                        AND TRACE TRAPPING ENABLED IS APPROXIMATELY 30 SECONDS.

                                3.0     ERROR INFORMATION
                                        -----------------

                                3.1     ERROR REPORTING PROCEDURES
                                        --------------------------

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

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

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

                                3.2     INTERPRETING ERROR REPORTS
                                        --------------------------

                                        EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH
                                        THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE
                                        ERROR CALL (ERRORPC).  THESE TWO VALUES PINPOINT THE
                                        PLACE IN THE CODE THAT THE ERROR OCCURRED.  BY REFERRING
                                        TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE
                                        COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST.
                                        A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING
                                        WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC
                                        AND FUNCTIONS BEING TESTED.

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

                                        BY USING THE COMMENTS AND TEST DESCRIPTION FOUND IN
                                        THE PROGRAM LISTING TO DETERMINE WHAT FUNCTION OR
```

LOGIC WAS BEING TESTED, THE OPERATOR CAN THEN REFER
TO THE ENGINEERING DRAWINGS TO ISOLATE THE PROBABLE
CAUSE FOR THE FAILURE.

3.3 SAMPLE ERROR REPORT

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

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

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

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

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

4.2 END-OF-PASS MESSAGE

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

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

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

4.3 T-BIT TRAPPING

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

REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE MESSAGE "POWER FAILURE-RESTARTING" IS TYPED OUT AND THE PROGRAM WILL RESTART EXECUTION AT "RESTRT:"

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.

```
                                  12 11 10 09 08 07 06 05 04 03 02 01 00
                                  ---------------------------------------
                                 / 0  1  1  1  1  1  1/ 1  1  1  1  1  0/  VBA*
                                 ---------------------------------------
                                                                    I
                                  (ADDED TO)                        I
                                                                    I
     15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00                I
     -----------------------------------------------                I
    / 0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  1/            I          PAR**
     -----------------------------------------------               I
                                    I                               I
                                    i                               I
                                    V                               V
     21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
     ---------------------------------------------------------------
    / 0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  1  1  1  1  1  0/ PBA
     ---------------------------------------------------------------
```

*= 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

4.6 RELOCATION THROUGHOUT MEMORY

A FEATURE WAS ADDED TO ALLOW THE CONSTRUCTION OF PHYSICAL BUS ADDRESSES ABOVE THE NORMAL 16K LIMIT. THE SETTING OF THE LOCATION $MADR1 IN THE E-TABLE WITH ONE OF THE FOLLOWING CONSTANTS WILL ACCESS LOCATIONS BETWEEN 0 AND 7600 OF EACH 4K GROUP UP TO THE MAXIMUM MEMORY ON THE SYSTEM. THE FIRST LOCATION OF EACH BLOCK(32 WORDS) IS WRITTEN AND READ. SEE TEST #24 IN THE LISTING FOR MORE DETAILS.

CONSTANT	MAX. MEM.	CONSTANT	MAX. MEM.	CONSTANT	MAX. MEM.
0 OR 600	16K	3200	56K	5600	96K
1000	20K	3400	60K	6000	100K

1200	24K	3600	64K	6200	104K
1400	28K	4000	68K	6400	108K
1600	32K	4200	72K	6600	112K
2000	36K	4400	76K	7000	116K
2200	40K	4600	80K	7200	120K
2400	44K	5000	84K	7400	124K
2600	48K	5200	88K	7600	128K
3000	52K	5400	92K		

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

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

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

5.2 PROGRAM LISTING

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

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

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

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

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH

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

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

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

```
                                .TITLE  MD-11-CJKDA-A F-11 MMU DIAG
                                ;*COPYRIGHT (C) JAN-79
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            INHIBIT TRACE TRAP
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>
                                .SBTTL  BASIC DEFINITIONS

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

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

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

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

```
                        PR2=    100             ;;PRIORITY LEVEL 2
                        PR3=    140             ;;PRIORITY LEVEL 3
                        PR4=    200             ;;PRIORITY LEVEL 4
                        PR5=    240             ;;PRIORITY LEVEL 5
                        PR6=    300             ;;PRIORITY LEVEL 6
                        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                  1BITVEC=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                  U1PDR0= 177600
177602                  U1PDR1= 177602
177604                  U1PDR2= 177604
177606                  U1PDR3= 177606
177610                  U1PDR4= 177610
177612                  U1PDR5= 177612
177614                  U1PDR6= 177614
177616                  U1PDR7= 177616

                        ;*USER "I" PAGE ADDRESS REGISTERS

177640                  U1PAR0= 177640
177642                  U1PAR1= 177642
177644                  U1PAR2= 177644
177646                  U1PAR3= 177646
177650                  U1PAR4= 177650
177652                  U1PAR5= 177652
177654                  U1PAR6= 177654
177656                  U1PAR7= 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
     000046  034056                           $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                  $UNITM: .WORD   5       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000216  000016                          .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
```

```
                        $LF:    .ASCIZ  <12>            ;;LINE FEED
                        ;;*****************************************************************
                        .SBTTL  APT MAILBOX-ETABLE

                        ;;*****************************************************************
                        .EVEN
001226                  $MAIL:                          ;;APT MAILBOX
001226  000000          $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001230  000000          $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001232  000000          $TESTN: .WORD   ATESTN  ;;TEST NUMBER
001234  000000          $PASS:  .WORD   APASS   ;;PASS COUNT
001236  000000          $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001240  000000          $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001242  000000          $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001244  000000          $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001246                  $ETABLE:                        ;;APT ENVIRONMENT TABLE
001246     000          $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001247     000          $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001250  000000          $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001252  000000          $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001254  000000          $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                        ;*                      BITS 15-11=CPU TYPE
                        ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                        ;*                              11/70=06,PDQ=07,Q=10
                        ;*                      BIT 10=REAL TIME CLOCK
                        ;*                      BIT  9=FLOATING POINT PROCESSOR
                        ;*                      BIT  8=MEMORY MANAGEMENT
001256     000          $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001257     000          $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                        ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                        ;*                              900 NSEC CORE=001
                        ;*                              300 NSEC BIPOLAR=002
                        ;*                              500 NSEC MOS=003
001260  000000          $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                        ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001262                  $ETEND:
                        .MEXIT

001262  000000         TESTNO: .WORD   0               ;HOLDS TEST NUMBER FOR TYPEOUTS
001264  000000         WASR6:  .WORD   0               ;USED TO STORE THE STACK POINTER AFTER A TRAP
001266  000000         TRAPPC: .WORD   0               ;USED TO STORE THE PC OF A TRAP OR ABORT
001270  000000         TRAPPS: .WORD   0               ;USED TO STORE THE PS OF A TRAP OR ABORT
001272  000000         WASSR0: .WORD   0               ;USED TO STORE CONTENTS OF SR0
001274  000000         WASSR2: .WORD   0               ;USED TO STORE CONTENTS OR SR2
001276  000000         TBITPS: .WORD   0               ;SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
001300  000000         ANDADR: .WORD   0               ;HOLDS RESULT OF ADDRESSES BEING AND-ED
001302  000000         ORADR:  .WORD   0               ;HOLDS RESULT OF ADDRESSES BEING OR-ED
001304  000000         TONUM:  .WORD   0               ;HOLDS NUMBER OF TIME-OUTS
001306  000000         VIRT1:  .WORD   0               ;HOLDS VIRTUAL ADDRESS TO BE CONVERTED
001310  000000         VIRT2:  .WORD   0               ;       "       "       "
001312  000000         PBALO:  .WORD   0               ;HOLDS BITS <15:00> OF PHYSICAL ADDRESS
001314  000000         PBAHI:  .WORD   0               ;HOLDS BITS <17:16> OF PHYSICAL ADDRESS
```

```
                                .SBTTL  ERROR POINTER TABLE

                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:       EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


001316                          $ERRTB:

                                ;*ITEM 1
001316  040714                          EM1                     ;UNEXPECTED CPU TRAP TO LOC. 004
001320  044005                          DH1                     ;OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC
001322  047212                          DT1                     ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERPPC, 0
001324  050022                          DF1                     ;0,0,0,0,0

                                ;*ITEM 2
001326  040754                          EM2                     ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
001330  044055                          DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC
001332  047226                          DT2                     ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
001334  050027                          DF2                     ;0,0,0,0,0,0,0

                                ;*ITEM 3
001336  041023                          EM3                     ;PRIORITY BITS SET WRONG IN PSW
001340  044145                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001342  047246                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001344  050036                          DF3                     ;0,0,0,0

                                ;*ITEM 4
001346  041062                          EM4                     ;MODE BITS SET WRONG IN PSW
001350  044145                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001352  047246                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001354  050036                          DF3                     ;0,0,0,0

                                ;*ITEM 5
001356  041115                          EM5                     ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
001360  044145                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001362  047246                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001364  050036                          DF3                     ;0,0,0,0

                                ;*ITEM 6
001366  041170                          EM6                     ;KERNEL R6 CHANGED BY WRITING USER R6
001370  044145                          DH3                     ;WROTE   READ    TESTNO  ERRORPC
001372  047246                          DT3                     ;$REG0,$REG1,TESTNO,$ERRPC,0
001374  050036                          DF3                     ;0,0,0,0

                                ;*ITEM 7
001376  041235                          EM7                     ;A MEMORY MGMT. REG. TIMED OUT
001400  044205                          DH7                     ;ADDRESS TESTNO  ERRORPC
001402  047260                          DT7                     ;$REG0,TESTNO,$ERRPC,0
```

```
001404  050042                  DF7             ;0,0,0

                        ;*ITEM 10
001406  041273                  EM10            ;SUMMARY OF MEM. MGMT. REG. TIMEOUTS
001410  044235                  DH10            ;REGISTER-ADDRS  NUM. OF
                                                ;AND-ED  OR-ED   TIMOUTS TESTNO ERRORPC
001412  047270                  DT10            ;ANDADR,ORADR,TONUM,TESTNO,$ERRPC,0
001414  050045                  DF10            ;0,0,1,0,0

                        ;*ITEM 11
001416  041337                  EM11            ;MEM. MGMT. REG. WOULD NOT CLEAR
001420  044335                  DH11            ;REGISTR READ    READ-(BINARY)
                                                ;ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
001422  047304                  DT11            ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
001424  050052                  DF11            ;0,0,2,0,0

                        ;*ITEM 12
001426  041377                  EM12            ;MEM. MGMT. REG. BITS NOT SET CORRECTLY
001430  044455                  DH12            ;REGISTR WROTE   READ    READ
                                                ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
001432  047320                  DT12            ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
001434  050057                  DF12            ;0,0,0,2,0,0

                        ;*ITEM 13
001436  041446                  EM13            ;SR0 EFFECTED BY WRITE TO PSW
001440  044615                  DH13            ;READ    TESTNO  ERRORPC
001442  047336                  DT13            ;$REG0,TESTNO,$ERRPC,0
001444  050065                  DF13            ;0,0,0

                        ;*ITEM 14
001446  041503                  EM14            ;SR1 DID NOT READ ALL ZEROS
001450  044615                  DH13            ;READ    TESTNO  ERRORPC
001452  047336                  DT13            ;$REG0,TESTNO,$ERRPC,0
001454  050065                  DF13            ;0,0,0

                        ;*ITEM 15
001456  041536                  EM15            ;DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
001460  044455                  DH12            ;REGISTER WROTE   READ    READ
                                                ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO  ERRORPC
001462  047320                  DT12            ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
001464  050057                  DF12            ;0,0,0,2,0,0

                        ;*ITEM 16
001466  041612                  EM16            ;DUAL ADDRESSING BETWEEN PAR-PDR'S
001470  044645                  DH16            ;PAR-PDR PAR-PDR
                                                ;CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
001472  047346                  DT16            ;$REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
001474  050070                  DF16            ;0,0,0,0,0,0

                        ;*ITEM 17
001476  041654                  EM17            ;PHYS. ADDR. FORMED READ WRONG
001500  044745                  DH17            ;PHYSICAL VIRTUAL
                                                ;ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC
001502  047364                  DT17            ;PBALO,VIRT1,$REG4,TESTNO,$ERRPC,0
001504  050076                  DF17            ;3,0,0,0,0
```

```
                                ;*ITEM 20
001506  041712                          EM20            ;PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
001510  045035                          DH20            ;PHYSICL PAR 4   PAR 5
                                                        ;ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO
001512  047400                          DT20            ;PBALO,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
001514  050103                          DF20            ;3,0,0,0,0,0,0,0

                                ;*ITEM 21
001516  041764                          EM21            ;W-BIT DID NOT GET SET IN PDR
001520  045163                          DH21            ;PDR     VIRTUAL
                                                        ;TESTED  ADDRESS TESTNO  ERRORPC
001522  047422                          DT21            ;$REG5,$REG3,TESTNO,$ERRPC,0
001524  050113                          DF21            ;0,0,0,0

                                ;*ITEM 22
001526  042021                          EM22            ;W-BIT SET IN MORE THAN ONE PDR
001530  045243                          DH22            ;PDR IN  PDR     VIRTUAL
                                                        ;ERROR   TESTED  ADDRESS TESTNO  ERRORPC
001532  047434                          DT22            ;$REG0,$REG5,$REG3,TESTNO,$ERRPC,0
001534  050117                          DF22            ;0,0,0,0,0

                                ;*ITEM 23
001536  042060                          EM23            ;W-BIT NOT CLEARED BY WRITING TO PDR
001540  045342                          DH23            ;PDR     TESTNO  ERRORPC
001542  047450                          DT23            ;$REG5,TESTNO,$ERRPC,0
001544  050124                          DF23            ;0,0,0

                                ;*ITEM 24
001546  042124                          EM24            ;WRITING SR0 SET W-BIT IN KIPDR7
001550  045372                          DH24            ;PDR WAS EXPECTD TESTNO  ERRORPC
001552  047460                          DT24            ;$REG2,$REG1,TESTNO,$ERRPC,0
001554  050127                          DF24            ;0,0,0,0

                                ;*ITEM 25
001556  042164                          EM25            ;W-BIT GOT SET DURING TIMEOUT ABORT
001560  045372                          DH24            ;PDR WAS EXPECTD TESTNO  ERRORPC
001562  047460                          DT24            ;$REG2,$REG1,TESTNO,$ERRPC,0
001564  050127                          DF24            ;0,0,0,0

                                ;*ITEM 26
001566  042227                          EM26            ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
001570  045432                          DH26            ;PDR 4   PSW     TESTNO  ERRORPC
001572  047472                          DT26            ;$REG2,$TMP0,TESTNO,$ERRPC,0
001574  050127                          DF24            ;0,0,0,0

                                ;*ITEM 27
001576  042277                          EM27            ;ACCESS ERROR DID NOT ABORT INSTRUCTION
001600  045432                          DH26            ;PDR 4   PSW     TESTNO  ERRORPC
001602  047472                          DT26            ;$REG2,$TMP0,TESTNO,$ERRPC,0
001604  050127                          DF24            ;0,0,0,0

                                ;*ITEM 30
001606  042346                          EM30            ;SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
001610  045472                          DH30            ;SR0 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
001612  047504                          DT30            ;WASSR0,$REG3,$REG2,$TMP0 TESTNO,$ERRPC,0
001614  050133                          DF30            ;0,0,0,0,0,0
```

```
1029
1030                                    ;*ITEM 31
1031  001616  042420                          EM31            ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1032  001620  045552                          DH31            ;SR2 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1033  001622  047522                          DT31            ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1034  001624  050133                          DF30            ;0,0,0,0,0,0
1035
1036                                    ;*ITEM 32
1037  001626  042465                          EM32            ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1038  001630  045632                          DH32            ;V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC
1039  001632  047540                          DT32            ;$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1040  001634  050133                          DF30            ;0,0,0,0,0,0
1041
1042                                    ;*ITEM 33
1043  001636  042546                          EM33            ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1044  001640  045712                          DH33            ;V.B.A.  KIPDR4  TESTNO  ERRORPC
1045  001642  047556                          DT33            ;$REG0,$REG4,TESTNO,$ERRPC,0
1046  001644  050127                          DF24            ;0,0,0,0
1047
1048                                    ;*ITEM 34
1049  001646  042631                          EM34            ;SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1050  001650  045752                          DH34            ;V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC
1051  001652  047570                          DT34            ;$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1052  001654  050133                          DF30            ;0,0,0,0,0,0
1053                                    ;*ITEM 35
1054  001656  042420                          EM31            ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1055  001660  046032                          DH35            ;V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC
1056  001662  047606                          DT35            ;$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1057  001664  050133                          DF30            ;0,0,0,0,0,0
1058
1059                                    ;*ITEM 36
1060  001666  042420                          EM31            ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1061  001670  046112                          DH36            ;SR2 WAS EXPECTD TESTNO  ERRORPC
1062  001672  047624                          DT36            ;WASSR2,$REG1,TESTNO,$ERRPC,0
1063  001674  050127                          DF24            ;0,0,0,0
1064
1065                                    ;*ITEM 37
1066  001676  042707                          EM37            ;SR0 OR SR2 CHANGED BY A SECOND ABORT
1067  001700  046152                          DH37            ;FIRST ABORT     SECOND ABORT
1068                                                            ;SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC
1069  001702  047636                          DT37            ;$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1070  001704  050133                          DF30            ;0,0,0,0,0,0
1071
1072                                    ;*ITEM 40
1073  001706  042754                          EM40            ;SR0 OR SR2 WAS NOT "RESET" BY A RESET
1074  001710  046267                          DH40            ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1075  001712  047654                          DT40            ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1076  001714  050127                          DF24            ;0,0,0,0
1077
1078                                    ;*ITEM 41
1079  001716  043023                          EM41            ;SR2 NOT TRACKING CORRECTLY
1080  001720  046112                          DH36            ;SR2 WAS EXPECTD TESTNO  ERROPC
1081  001722  047624                          DT36            ;WASSR2,$REG1,TESTNO,$ERRPC,0
1082  001724  050127                          DF24            ;0,0,0,0
1083
1084                                    ;*ITEM 42
```

```
1085  001726  043056                    EM42          ;DID NOT TRAP THRU KERNEL SPACE
1086  001730  046327                    DH42          ;PSW WAS R6 WAS  TESTNO  ERRORPC
1087  001732  047666                    DT42          ;$REG1,$REG2,TESTNO,$ERRPC,0
1088  001734  050127                    DF24          ;0,0,0,0
1089
1090                              ;*ITEM 43
1091  001736  043115                    EM43          ;KT ERROR NOT SERVICED ON TIMEOUT ERROR
1092  001740  045342                    DH23          ;PDR     TESTNO  ERRORPC
1093  001742  047450                    DT23          ;$REG5,TESTNO,$ERRPC,0
1094  001744  050124                    DF23          ;0,0,0
1095
1096                              ;*ITEM 44
1097  001746  043164                    EM44          ;SR0 OR SR2 CHANGED BY TIMEOUT ERROR
1098  001750  046367                    DH44          ;EXPECTED         RECEIVED
1099                                                   ;SR0     SR2      SR0 WAS SR2 WAS TESTNO  ERRORPC
1100  001752  047700                    DT44          ;$REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1101  001754  050133                    DF30          ;0,0,0,0,0,0
1102
1103                              ;*ITEM 45
1104  001756  043230                    EM45          ;ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)
1105  001760  046501                    DH45          ;EXPECTED:
1106                                                   ;PSW     PC        SR0      SR2
1107                                                   ;170017  (3$+4)  020147  (3$)
1108                                                   ;RECEIVED
1109                                                   ;PSW     PC        SR0      SR2      TESTNO  ERRORPC
1110  001762  047716                    DT45          ;$REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1111  001764  050133                    DF30          ;0,0,0,0,0,0
1112
1113                              ;*ITEM 46
1114  001766  043303                    EM46          ;MFPI INSTRUCTION PUSHED WRONG DATA
1115  001770  046676                    DH46          ;DATA    DATA
1116                                                   ;EXPECTD RECEIVD TESTNO  ERRORPC
1117  001772  047734                    DT46          ;$REG0,$REG1,TESTNO,$ERRPC,0
1118  001774  050141                    DF46          ;0,0,0,0
1119
1120                              ;*ITEM 47
1121  001776  043346                    EM47          ;MTPI INSTRUCTION LOADED WRONG DATA
1122  002000  046676                    DH46          ;DATA    DATA
1123                                                   ;EXPECTD RECEIVD TESTNO  ERRORPC
1124  002002  047734                    DT46          ;$REG0,$REG1,TESTNO,$ERRPC,0
1125  002004  050141                    DF46          ;0,0,0,0
1126
1127                              ;*ITEM 50
1128  002006  043411                    EM50          ;STACK NOT PUSHED BY MFPI-MTPI
1129  002010  046753                    DH50          ;TESTNO  ERRORPC
1130  002012  047746                    DT50          ;TESTNO,$ERRPC,0
1131  002014  050145                    DF50          ;0,0
1132
1133                              ;*ITEM 51
1134  002016  043447                    EM51          ;KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1135  002020  046773                    DH51          ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1136  002022  047754                    DT51          ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1137  002024  050147                    DF51          ;0,0,0,0
1138
1139                              ;*ITEM 52
1140  002026  043525                    EM52          ;WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE
```

```
1141  002030  047033                    DH52            ;PHYSICL PAR 4
1142                                                    ;ADDRESS V.B.A.  PAR 4   SR0 WAS SR2 WAS PSW    TESTNO
1143  002032  047766                    DT52            ;PBALO,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
1144  002034  050153                    DF52            ;3,0,0,0,0,0,0,0
1145                              ;*ITEM 53
1146  002036  043603                    EM53            ;MFPD INSTRUCTION PUSHED WRONG DATA
1147  002040  046676                    DH46            ;DATA    DATA
1148                                                    ;EXPECTD RECEIVD TESTNO  ERRORPC
1149  002042  047734                    DT46            ;$REG0,$REG1,TESTNO,$ERRPC,0
1150  002044  050141                    DF46            ;0,0,0,0
1151
1152                              ;*ITEM 54
1153  002046  043646                    EM54            ;STACK NOT PUSHED BY MFPD-MTPD
1154  002050  046753                    DH50            ;TESTNO  ERRORPC
1155  002052  047746                    DT50            ;TESTNO,$ERRPC,0
1156  002054  050145                    DF50            ;0,0
1157
1158                              ;*ITEM 55
1159  002056  043704                    EM55            ;PAR OR PDR WAS CHANGED BY A RESET
1160  002060  044335                    DH11            ;REGISTR READ    READ-(BINARY)
1161                                                    ;ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC
1162  002062  047304                    DT11            ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
1163  002064  050052                    DF11            ;0,0,2,0,0
1164
1165                              ;*ITEM 56
1166  002066  043742                    EM56            ;PSW CHANGED BY AN RTI IN USER MODE
1167  002070  047151                    DH56            ;PSW WAS EXPECTD TESTNO  ERRORPC
1168  002072  050010                    DT56            ;$REG1,$REG2,TESTNO,$ERRPC,0
1169  002074  050163                    DF56            ;0,0,0,0
1170
1171
```

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

                                        .SBTTL  CPU TRAP HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
                                        ;*      "ERRVEC" (LOC. 004).  IF THIS SUBROUTINE IS ENTERED BY A
                                        ;*      SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
                                        ;*      EXECUTED.
                                        ;*
                                        ;;*****************************************************************
002076  005227                  TIMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
002100  177777                  TIMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
002102  001403                          BEQ     1$              ;BRANCH IF FIRST TIME IN
002104  005237  001226                  INC     $MSGTYPE        ;TELL APT THERE WAS AN ERROR
002110  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE I FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
002112  012637  001266          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
002116  012637  001270                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
002122  010637  001264                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
002126  104001                          ERROR   1               ;UNEXPECTED TRAP OR ABORT TO LOC. 4
002130  012737  177777  002100          MOV     #-1,TIMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
002136  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
002142  013746  001266                  MOV     TRAPPC,-(KSP)
002146  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT


                                        .SBTTL  MEMORY MANAGEMENT TRAP HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
                                        ;*      TRAPS AND ABORTS THRU "MMVEC" (LOC. 250).  IF THIS SUBROUTINE IS
                                        ;*      ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
                                        ;*      HALT IS EXECUTED.
                                        ;*
                                        ;;*****************************************************************
002150  005227                  MGMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
002152  177777                  MGMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
002154  001403                          BEQ     1$              ;BRANCH IF FIRST TIME IN
002156  005237  001226                  INC     $MSGTYPE        ;TELL APT THERE WAS AN ERROR
002162  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE I FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
002164  012637  001266          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
002170  012637  001270                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
002174  010637  001264                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
002200  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE CONTENTS OF KT STATUS REG. 0
002206  013737  177576  001274          MOV     SR2,WASSR2      ;SAVE CONTENTS OF KT STATUS REG. 2
002214  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN STATUS REG 0
002222  104002                          ERROR   2               ;UNEXPECTED TRAP OR ABORT TO LOC. 250
002224  012737  177777  002152          MOV     #-1,MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
```

```
 1228  002232  013746  001270                  MOV     TRAPPS,-(KSP)   ;PU: PC & PS OF TRAP ON STACK
 1229  002236  013746  001266                  MOV     TRAPPC,-(KSP)
 1230  002242  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT.
 1231
```

```
1232                                            .SBTTL
1233                                            .SBTTL  *****   STARTING POINT OF TEST  *****
1234                                            .SBTTL  *****   STARTING ADDRESS OF 200 *****
1235            020000                          .=20000
1236
1237  020000                                START:
1238                                          .SBTTL  INITIALIZE THE COMMON TAGS
1239                                          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1240  020000  012706  001100                         MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1241  020004  005026                                 CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1242  020006  022706  001140                         CMP     #SWR,R6 ;;DONE?
1243  020012  001374                                 BNE     .-6             ;;LOOP BACK IF NO
1244  020014  012706  001100                         MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1245                                          ;;INITIALIZE A FEW VECTORS
1246  020020  012737  034136  000020                 MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1247  020026  012737  000340  000022                 MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1248  020034  012737  034416  000030                 MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1249  020042  012737  000340  000032                 MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1250  020050  012737  040400  000034                 MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1251  020056  012737  000340  000036                 MOV     #340,@#TRAPVEC+2;LEVEL 7
1252  020064  012737  040466  000024                 MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1253  020072  012737  000340  000026                 MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1254  020100  013737  033704  033676                 MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1255  020106  005037  001212                         CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1256  020112  005037  001214                         CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1257  020116  112737  000001  001115                 MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1258                                          ;;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "$RTRN", IN
1259                                          ;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT".
1260  020124  012737  034122  000014                 MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
1261  020132  012737  000340  000016                 MOV     #340,@#TBITVEC+2 ;;LEVEL 7
1262  020140  012737  000002  034122                 MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
1263  020146  012737  020174  000010                 MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
1264  020154  005046                                 CLR     -(SP)           ;;DUMMY PS
1265  020156  012746  020164                         MOV     #64$,-(SP)      ;;AND PC
1266  020162  000006                                 RTT                     ;;TRY THE RTT
1267  020164  012737  000006  034122  64$:    MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
1268  020172  000402                                 BR      66$
1269  020174  062706  000010          65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
1270  020200  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
1271  020206  005037  034130                         CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
1272  020212  012737  020212  001106                 MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1273  020220  012737  020220  001110                 MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1274                                          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1275                                          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1276  020226  013746  000004                         MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1277  020232  012737  020266  000004                 MOV     #67$,@#ERRVEC   ;;SET UP ERROR VECTOR
1278  020240  012737  177570  001140                 MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1279  020246  012737  177570  001142                 MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1280  020254  022777  177777  160656                 CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1281  020262  001012                                 BNE     69$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1282                                                                          ;;AND  THE HARDWARE SWR IS NOT = -1
1283  020264  000403                                 BR      68$             ;;BRANCH IF NO TIMEOUT
1284  020266  012716  020274          67$:    MOV     #68$,(SP)       ;;SET UP FOR TRAP RETURN
1285  020272  000002                                 RTI
1286  020274  012737  000176  001140  68$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1287  020302  012737  000174  001142                 MOV     #DISPREG,DISPLAY
```

```
1288  020310  012637  000004          69$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1289
1290  020314  005037  001234                  CLR     $PASS           ;;CLEAR PASS COUNT
1291  020320  132737  000200  001247          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
1292  020326  001403                          BEQ     70$             ;;YES,USE NON-APT SWITCH
1293  020330  012737  001250  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
1294  020336                                  70$:
1295                                          .SBTTL  TYPE PROGRAM NAME
1296                                          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1297  020336  005227  177777                  INC     #-1             ;;FIRST TIME?
1298  020342  001047                          BNE     71$             ;;BRANCH IF NO
1299  020344  022737  034056  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
1300  020352  001443                          BEQ     71$             ;;BRANCH IF YES
1301  020354  104401  020422                  TYPE    ,72$            ;;TYPE ASCIZ STRING
1302                                          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1303  020360  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1304  020364  001012                          BNE     73$             ;;BRANCH IF YES
1305  020366  123727  001246  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
1306  020374  001406                          BEQ     73$             ;;BRANCH IF YES
1307  020376  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1308  020404  001005                          BNE     74$             ;;BRANCH IF NO
1309  020406  104407                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1310  020410  000403                          BR      74$
1311  020412  112737  000001  001134  73$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1312  020420                                  74$:
1313  020420  000420                          BR      71$             ;;GET OVER THE ASCIZ
1314                                  ;;72$:  .ASCIZ  <CRLF>#MD-11-CJKDA-A F-11 MMU DIAG.#<CRLF>
1315  020462                                  71$:
1316
1317  020462                                  RESTRT:
1318
1319                                          ;;*****************************************************************
1320                                          ;THE FOLLOWING TURNS ON A TIMER ON THE MULTI-OPTION TESTER
1321                                          ;IN MANUFACTORING.
1322
1323  020462  012737  020504  000004  TIMEON: MOV     #LOOP,@#4       ;SETUP RETURN ADDRESS IN CASE OF TIMEOUT
1324  020470  012737  000340  000006          MOV     #340,@#6        ;SETUP PSW
1325  020476  012737  000002  164000          MOV     #2,@#164000     ;SET START BIT IN MULTI-TESTER
1326                                          ;;*****************************************************************
1327
1328  020504  012706  001100          LOOP:   MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
1329  020510  012737  002076  000004          MOV     #TIMERR,ERRVEC  ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1330  020516  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1331  020524  012737  002150  000250          MOV     #MGMERR,MMVEC   ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1332  020532  012737  000340  000252          MOV     #340,MMVEC+2    ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1333  020540  012700  177777                  MOV     #-1,R0          ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1334  020544  010037  002100                  MOV     R0,TIMFLG       ;INITIALIZE CPU ERROR FLAG
1335  020550  010037  002152                  MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1336  020554  012737  000340  001276          MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1337  020562  005037  177572                  CLR     SR0             ;BE SURE MEM. MGMT IS OFF TO START WITH
1338
```

```
                                ;;*****************************************************************
                                ;*TEST 1        PSW PRIORITY BIT TEST
                                ;*
                                ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <7:5> "PRIORITY BITS"
                                ;*      TO SEE THAT SOME OF THE BASIC "DATA PATH" LOGIC IS WORKING.
                                ;*
                                ;;*****************************************************************
020566  000004                  TST1:   SCOPE
020570  012737  020600  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
020576  005000                          CLR     R0              ;INITIALIZE R0 WITH PRIORITY=0 DATA
020600  005001                  2$:     CLR     R1              ;PREPARE R1 TO ACCEPT DATA READ
020602  106400                          MTPS    R0              ;WRITE PRIORITY BITS IN THE PSW
020604  106701                          MFPS    R1              ;READ BACK THE LOW BYTE OF PSW
020606  042701  177437                  BIC     #177437,R1      ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
020612  020001                          CMP     R0,R1           ;WAS CORRECT PRIORITY SET IN THE PSW?
020614  001401                          BEQ     3$              ;BRANCH IF YES
020616  104003                          ERROR   3               ;PRIORITY BITS SET WRONG IN PSW
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000770
020620  062700  000040          3$:     ADD     #40,R0          ;CHANGE DATA TO NEXT PRIORITY
020624  022700  000400                  CMP     #400,R0         ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
020630  001363                          BNE     2$              ;BRANCH IF NO
020632  012737  020570  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                                ;;*****************************************************************
                                ;*TEST 2        PSW MODE BIT TEST
                                ;*      THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> "MODE BITS"
                                ;*
                                ;;*****************************************************************
020640  000004                  TST2:   SCOPE
020642  012737  020652  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
020650  005000                          CLR     R0              ;INITIALIZE R0 WITH MODE BITS = 0000
020652  005037  177776          2$:     CLR     PSW             ;INITIALIZE PSW
020656  050037  177776                  BIS     R0,PSW          ;BIT SET THE PSW MODE BITS WITH R0
020662  013701  177776                  MOV     PSW,R1          ;READ BACK THE CONTENTS OF THE PSW
020666  042701  007777                  BIC     #007777,R1      ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
020672  020001                          CMP     R0,R1           ;WERE THE MODE BITS SET CORRECTLY?
020674  001403                          BEQ     3$              ;BRANCH IF YES
020676  005037  177776                  CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
020702  104004                          ERROR   4               ;MODE BITS SET WRONG IN PSW
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH,
                                                                ;"BR 2$" = 000763
020704  062700  010000          3$:     ADD     #10000,R0       ;CHANGE MODE BIT DATA
020710  001360                          BNE     2$              ;BRANCH IF STILL MORE COMBINATIONS
020712  012737  020642  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
020720  005037  177776                  CLR     PSW             ;RESET PSW BEFORE LEAVING

                                ;;*****************************************************************
                                ;*TEST 3        BYTE ADDRESSING TEST FOR PSW
                                ;*
                                ;*      THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
                                ;*      AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
```

```
                                        ;*
                                        ;;*****************************************************************
020724  000004                  TST3:   SCOPE
020726  012737  020734  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
020734  005037  177776          2$:     CLR     PSW             ;CLEAR THE PSW
020740  012700  000360                  MOV     #360,R0         ;PUT THE HIGH BYTE DATA INTO R0
020744  110037  177777                  MOVB    R0,PSW+1        ;WRITE THE HIGH BYTE OF THE PSW
020750  013701  177776                  MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
020754  042701  007437                  BIC     #007437,R1      ;MASK OFF THE T & CC BITS
020760  000300                          SWAB    R0              ;GET DATA WRITTEN IN HIGH BYTE OF R0
020762  020001                          CMP     R0,R1           ;WAS THE PSW WRITTEN TO CORRECTLY
020764  001403                          BEQ     3$              ;BRANCH IF YES
020766  005037  177776                  CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
020772  104005                          ERROR   5               ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000760
020774  012737  021002  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
021002  005037  177776          4$:     CLR     PSW             ;CLEAR THE PSW
021006  012700  000340                  MOV     #340,R0         ;PUT THE LOW BYTE DATA INTO R0
021012  110037  177776                  MOVB    R0,PSW          ;WRITE THE LOW BYTE OF THE PSW
021016  013701  177776                  MOV     PSW,R1          ;READ BACK THE ENTIRE PSW
021022  042701  007437                  BIC     #007437,R1      ;MASK OFF THE T&CC BITS
021026  020001                          CMP     R0,R1           ;WAS PSW WRITTEN TO CORRECTLY
021030  001403                          BEQ     5$              ;BRANCH IF YES
021032  005037  177776                  CLR     PSW             ;CLEAR PSW FOR ERROR REPORT
021036  104005                          ERROR   5               ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000736
021040  012737  020726  001110  5$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                                        ;;*****************************************************************
                                        ;*TEST 4        TEST AND SETUP OF STACK POINTERS
                                        ;*
                                        ;*      THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
                                        ;*      REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
                                        ;*      EACH OTHER.  KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
                                        ;*      KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
                                        ;*
                                        ;;*****************************************************************
021046  000004                  TST4:   SCOPE
021050  005037  177776                  CLR     PSW             ;GO TO KERNEL MODE
021054  012706  001100                  MOV     #KERSTK,KSP     ;SET KERNEL STACK POINTER TO 1100
021060  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
021066  012706  000700                  MOV     #USESTK,USP     ;SET USER STACK POINTER TO 700
021072  005037  177776                  CLR     PSW             ;BACK TO KERNEL MODE
021076  022706  001100                  CMP     #KERSTK,KSP     ;IS KERNEL R6 STILL 1100?
021102  001404                          BEQ     TST5            ;;BRANCH IF KERNEL R6 IS OKAY
021104  012700  001100                  MOV     #KERSTK,R0      ;SAVE DATA WRITTEN FOR ERROR REPORT
021110  010601                          MOV     KSP,R1          ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
021112  104006                          ERROR   6               ;KERNEL R6 CHANGED BY WRITING USER R6
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;000756
```

```
                                ;;*****************************************************************
                                ;*
                                ;*      THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
                                ;*      MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,SR3,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,SR3 TIMEOUT TEST
                                ;*
                                ;*      THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
                                ;*      0,1,2, AND 3.  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.
                                ;*
                                ;;*****************************************************************
021114  000004                  TST5:   SCOPE

021116  012737  021160  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
021124  012737  021222  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
021132  012700  177572                  MOV     #SR0,R0         ;LOAD R0 WITH ADDRESS OF FIRST REG.
021136  012701  000003                  MOV     #3,R1           ;LOAD R1 WITH THE LOOP COUNT
021142  012737  177777  001300          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDRS. LOC.
021150  005037  001302                  CLR     ORADR           ;INITIALIZE "OR" OF ADDRS. LOC.
021154  005037  001304                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
021160  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A STATUS REGISTER
                                                                ;IF IT TIMES OUT GO TO 5$
021162  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT ADDRESS IN R0
021166  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
021170  005737  172516                  TST     @#172516        ;CHECK SR3 FOR RESPONSE
                                                                ;IF IT TIMES OUT GO TO 5$
021174  012737  021116  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
021202  005737  001304                  TST     TONUM           ;DID ANY OF THE STATUS REG.S TIMEOUT?
021206  001401                          BEQ     4$              ;BRANCH IF NO
021210  104010                          ERROR   10              ;SUMMARY OF STATUS REG. TIMEOUTS
021212  012737  002076  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021220  000414                          BR      TST6            ;;GO TO NEXT TEST

021222  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
021226  104007                          ERROR   7               ;ONE OF THE STATUS REGS. TIMED OUT
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000756
021230  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021232  050237  001302                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
021236  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
021240  040237  001300                  BIC     R2,ANDADR
021244  005237  001304                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
```

```
1507  021250  000744                          BR      3$              ;BRANCH BACK TO TEST THE NEXT ADDR.
1508
1509                                  ;;*****************************************************************
1510                                  ;*TEST 6        KERNEL PAR'S TIMEOUT TEST
1511                                  ;*
1512                                  ;*      THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1513                                  ;*      REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1514                                  ;*      RESPONDS TO THEIR ADDRESSES.
1515                                  ;*
1516                                  ;;*****************************************************************
1517  021252  000004                  1ST6:   SCOPE
1518
1519  021254  012737  021316  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1520  021262  012737  021354  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
1521  021270  012700  172340                  MOV     #KIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1522  021274  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
1523  021300  012737  177777  001300          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1524  021306  005037  001302                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
1525  021312  005037  001304                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1526  021316  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A KIPAR
1527                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1528  021320  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT KIPAR ADDRESS IN R0
1529  021324  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1530  021326  012737  021254  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1531  021334  005737  001304                  TST     TONUM           ;DID ANY OF THE KIPARS TIME OUT?
1532  021340  001401                          BEQ     4$              ;BRANCH IF NO
1533  021342  104010                          ERROR   10              ;SUMMARY OF KIPAR TIMEOUTS
1534  021344  012737  002076  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1535  021352  000414                          BR      TST7            ;;GO TO NEXT TEST
1536
1537  021354  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
1538  021360  104007                          ERROR   7               ;ONE OF THE KIPARS TIMED OUT
1539                                                                  ;FOR TIGHTER SCOPE LOOP
1540                                                                  ;REPLACE ERROR CALL WITH
1541                                                                  ;"BR 2$" = 000756
1542  021362  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1543  021364  050237  001302                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1544  021370  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1545  021372  040237  001300                  BIC     R2,ANDADR
1546  021376  005237  001304                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
1547  021402  000746                          BR      3$              ;BRANCH BACK TO TEST THE NEXT KIPAR
1548
1549                                  ;;*****************************************************************
1550                                  ;*TEST 7        KERNEL PDR'S TIMEOUT TEST
1551                                  ;*
1552                                  ;*      THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1553                                  ;*      REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1554                                  ;*      RESPONDS TO THEIR ADDRESSES.
1555                                  ;*
1556                                  ;;*****************************************************************
1557  021404  000004                  TST7:   SCOPE
1558
1559  021406  012737  021450  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1560  021414  012737  021506  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
1561  021422  012700  172300                  MOV     #KIPDR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1562  021426  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
```

```
1563  021432  012737  177777  001300          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1564  021440  005037  001302                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
1565  021444  005037  001304                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1566  021450  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A KIPDR
1567                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1568  021452  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT KIPDR ADDRESS IN R0
1569  021456  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1570  021460  012737  021406  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1571  021466  005737  001304                  TST     TONUM           ;DID ANY OF THE KIPDRS TIME OUT?
1572  021472  001401                          BEQ     4$              ;BRANCH IF NO
1573  021474  104010                          ERROR   10              ;SUMMARY OF KIPDR TIMEOUTS
1574  021476  012737  002076  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1575  021504  000414                          BR      TST10           ;;GO TO NEXT TEST
1576
1577  021506  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
1578  021512  104007                          ERROR   7               ;ONE OF THE KIPDRS TIMED OUT
1579                                                                  ;FOR TIGHTER SCOPE LOOP
1580                                                                  ;REPLACE ERROR CALL WITH
1581                                                                  ;"BR 2$" = 000756
1582  021514  010002                          MOV     R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1583  021516  050237  001302                  BIS     R2,ORADR        ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1584  021522  005102                          COM     R2              ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1585  021524  040237  001300                  BIC     R2,ANDADR
1586  021530  005237  001304                  INC     TONUM           ;INCREMENT THE TIMEOUT COUNTER
1587  021534  000746                          BR      3$              ;BRANCH BACK TO TEST THE NEXT KIPDR
1588
1589                                  ;;**************************************************************
1590                                  ;*TEST 10       USER PAR'S TIMEOUT TEST
1591                                  ;*
1592                                  ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1593                                  ;*      REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1594                                  ;*      RESPONDS TO THEIR ADDRESSES.
1595                                  ;*
1596                                  ;;**************************************************************
1597  021536  000004                  TST10:  SCOPE
1598
1599  021540  012737  021602  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1600  021546  012737  021640  000004          MOV     #5$,@#4         ;SET TIMEOUT VECTOR TO 5$
1601  021554  012700  177640                  MOV     #UIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1602  021560  012701  000010                  MOV     #10,R1          ;LOAD R1 WITH LOOP COUNT (8)
1603  021564  012737  177777  001300          MOV     #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1604  021572  005037  001302                  CLR     ORADR           ;INITIALIZE "OR" OF ADDR. LOC.
1605  021576  005037  001304                  CLR     TONUM           ;INITIALIZE "TIMEOUTS" COUNTER
1606  021602  005710                  2$:     TST     (R0)            ;TRY ADDRESSING A UIPAR
1607                                                                  ;IF IT TIMES OUT, WILL GO TO 5$
1608  021604  062700  000002          3$:     ADD     #2,R0           ;PUT NEXT UIPAR ADDRESS IN R0
1609  021610  077104                          SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
1610  021612  012737  021540  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1611  021620  005737  001304                  TST     TONUM           ;DID ANY OF THE UIPARS TIME OUT?
1612  021624  001401                          BEQ     4$              ;BRANCH IF NO
1613  021626  104010                          ERROR   10              ;SUMMARY OF UIPAR TIMEOUTS
1614  021630  012737  002076  000004  4$:     MOV     #TIMERR,@#4     ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1615  021636  000414                          BR      TST11           ;;GO TO NEXT TEST
1616
1617  021640  062706  000004          5$:     ADD     #4,KSP          ;CLEAN UP THE STACK
1618  021644  104007                          ERROR   7               ;ONE OF THE UIPARS TIMED OUT
```

```
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 2$" = 000756
021646  010002                  MOV     R0,R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021650  050237  001302            BIS     R2,ORADR      ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
021654  005102                  COM     R2            ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
021656  040237  001300            BIC     R2,ANDADR
021662  005237  001304            INC     TONUM         ;INCREMENT THE TIMEOUT COUNTER
021666  000746                  BR      3$            ;BRANCH BACK TO TEST THE NEXT UIPAR

                        ;;*************************************************************
                        ;*TEST 11       USER PDR'S TIMEOUT TEST
                        ;*
                        ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
                        ;*      REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
                        ;*      RESPONDS TO THEIR ADDRESSES.
                        ;*
                        ;;*************************************************************
021670  000004         TST11:  SCOPE

021672  012737  021734  001110  1$:     MOV     #2$,$LPERR    ;SET LOOP ON ERROR POINTER TO 2$
021700  012737  021772  000004          MOV     #5$,@#4       ;SET TIMEOUT VECTOR TO 5$
021706  012700  177600                  MOV     #UIPDR0,R0    ;LOAD R0 WITH ADDRESS OF FIRST REG.
021712  012701  000010                  MOV     #10,R1        ;LOAD R1 WITH LOOP COUNT (8)
021716  012737  177777  001300          MOV     #-1,ANDADR    ;INITIALIZE "AND" OF ADDR. LOC
021724  005037  001302                  CLR     ORADR         ;INITIALIZE "OR" OF ADDR. LOC.
021730  005037  001304                  CLR     TONUM         ;INITIALIZE "TIMEOUTS" COUNTER
021734  005710                  2$:     TST     (R0)          ;TRY ADDRESSING A UIPDR
                                                        ;IF IT TIMES OUT, WILL GO TO 5$
021736  062700  000002          3$:     ADD     #2,R0         ;PUT NEXT UIPDR ADDRESS IN R0
021742  077104                  SOB     R1,2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
021744  012737  021672  001110          MOV     #1$,$LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
021752  005737  001304                  TST     TONUM         ;DID ANY OF THE UIPDRS TIME OUT?
021756  001401                  BEQ     4$            ;BRANCH IF NO
021760  104010                  ERROR   10            ;SUMMARY OF UIPDR TIMEOUTS
021762  012737  002076  000004  4$:     MOV     #TIMERR,@#4   ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021770  000414                  BR      TST12         ;;GO TO NEXT TEST

021772  062706  000004          5$:     ADD     #4,KSP        ;CLEAN UP THE STACK
021776  104007                  ERROR   7             ;ONE OF THE UIPDRS TIMED OUT
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 2$" = 000756
022000  010002                  MOV     R0,R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
022002  050237  001302            BIS     R2,ORADR      ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
022006  005102                  COM     R2            ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
022010  040237  001300            BIC     R2,ANDADR
022014  005237  001304            INC     TONUM         ;INCREMENT THE TIMEOUT COUNTER
022020  000746                  BR      3$            ;BRANCH BACK TO TEST THE NEXT UIPDR

                        ;;*************************************************************
                        ;*TEST 12       SR0(15:13) BIT TEST & SR2 TEST
                        ;*
                        ;*      THIS TEST CHECKS BITS <15:13> OF STATUS REGISTER 0 TO SEE
                        ;*      THAT EACH CAN BE SET AND CLEARED AND THAT A "RESET" WILL
                        ;*      CLEAR ALL OF THEM.
```

```
1675                                    ;*      THE REST OF BITS IN SR0 WILL BE CHECKED LATER.
1676                                    ;*      ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
1677                                    ;*      OFF BUT LOCKS UP WHEN ANY OF SR0 ERROR BITS SET.
1678                                    ;*
1679                                    ;;*****************************************************************
1680  022022  000004                    TST12:  SCOPE
1681
1682  022024  012700  177572            1$:     MOV     #SR0,R0         ;LOAD ADDRESS OF SR0 INTO R0
1683  022030  012710  160000                    MOV     #160000,(R0)    ;SET BITS <15:13> IN SR0 (ERROR BITS)
1684  022034  000005                            RESET                   ;ISSUE AND "INIT" SIGNAL
1685  022036  011001                            MOV     (R0),R1         ;READ SR0 INTO R1 TO SEE IF CLEAR
1686  022040  001404                            BEQ     2$              ;BRANCH IF SR0<15:13> CLEARED BY "INIT"
1687  022042  104011                            ERROR   11              ;SR0<15:13> NOT CLEARED BY A 'RESET"
1688                                                                    ;FOR TIGHTER SCOPE LOOP
1689                                                                    ;REPLACE ERROR CALL WITH
1690                                                                    ;"BR 1$" = 000770
1691  022044  012737  022052  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1692  022052  013737  177576  001274  2$:     MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
1693  022060  012701  022052                    MOV     #2$,R1          ;LOAD EXPECTED CONTENTS INTO R1
1694  022064  020137  001274                    CMP     R1,WASSR2       ;IS SR2 TRACKING?
1695  022070  001401                            BEQ     3$              ;BRANCH IF YES
1696  022072  104041                            ERROR   41              ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
1697                                                                    ;FOR TIGHTER SCOPE LOOP
1698                                                                     REPLACE ERROR CALL WITH
1699                                                                    ;"BR 2$" = 000767
1700  022074  012737  022112  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
1701  022102  012701  100000                    MOV     #BIT15,R1       ;PUT DATA TO BE WRITTEN IN R1
1702  022106  012703  000003                    MOV     #3,R3           ;SETUP R3 AS A LOOP COUNTER
1703  022112  005010                    4$:     CLR     (R0)            ;CLEAR SR0
1704  022114  050110                    5$:     BIS     R1,(R0)         ;SET ONE OF THE ERROR BITS IN SR0
1705  022116  011002                            MOV     (R0),R2         ;READ SR0 INTO R2
1706  022120  020102                            CMP     R1,R2           ;DID RIGHT ERROR BIT GET SET?
1707  022122  001401                            BEQ     6$              ;BRANCH IF YES
1708  022124  104012                            ERROR   12              ;BITS WERE SET WRONG IN SR0
1709                                                                    ;FOR TIGHTER SCOPE LOOP
1710                                                                    ;REPLACE ERROR CALL WITH
1711                                                                    ;"BR 4$" = 000772
1712  022126  012704  022114            6$:     MOV     #5$,R4          ;LOAD EXPECTED CONTENTS OF SR2 IN R4
1713  022132  013737  177576  001274            MOV     SR2,WASSR2      ;READ SR2
1714  022140  020437  001274                    CMP     R4,WASSR2       ;DID SR2 LOCK UP WHEN ERROR
1715                                                                    ;BIT SET IN SR1?
1716  022144  001401                            BEQ     7$              ;BRANCH IF YES
1717  022146  104064                            ERROR   64              ;SR2 DID NOT LOCK UP
1718                                                                    ;FOR TIGHTER SCOPE LOOP
1719                                                                    ;REPLACE ERROR CALL WITH
1720                                                                    ;"BR 4$" = 000761
1721  022150  006001                    7$:     ROR     R1              ;CHANGE DATA TO CHECK NEXT ERROR BIT
1722  022152  077321                            SOB     R3,4$           ;LOOP BACK UNTIL <15:13> ALL TESTED
1723  022154  005010                            CLR     (R0)            ;CLEAR SR0 BEFORE LEAVING
1724  022156  012737  022024  001110            MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1725
1726                                    ;;*****************************************************************
1727                                    ;*TEST 13       SR0 & PSW DUAL ADDRESSING TEST
1728                                    ;*
1729                                    ;*      THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
1730                                    ;*      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.
                        ;*
                        ;;*****************************************************************
022164  000004         TST13:  SCOPE

022166  005037  177776  1$:     CLR     PSW             ;CLEAR THE PSW
022172  005037  177572          CLR     SR0             ;CLEAR STATUS REGISTER 0
022176  106427  000340          MTPS    #340            ;SET PRIORITY 7 IN LOW BYTE OF PSW
022202  013700  177572          MOV     SR0,R0          ;READ STATUS REGISTER 0
022206  001401                  BEQ     2$              ;BRANCH IF IT WAS STILL 0
022210  104013                  ERROR   13              ;SR0 EFFECTED BY A WRITE TO THE PSW
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 1$" = 000767
022212  005037  177572  2$:     CLR     SR0             ;BE SURE SR0 IS 0 BEFORE LEAVING
022216  005037  177776          CLR     PSW             ;BE SURE PSW IS 0 BEFORE LEAVING

                        ;;*****************************************************************
                        ;*TEST 14       TEST THAT SR1 READS ALL ZEROS
                        ;*
                        ;*      THIS TESTS CHECKS THAT STATUS REGISTER 1
                        ;*      RESPONDS WITH ALL ZEROS, AND THAT ONLY BITS<5:4>
                        ;*      OF STATUS REGISTER 3 ARE WRITEABLE.
                        ;*
                        ;;*****************************************************************
022222  000004         TST14:  SCOPE
022224  012700  177777  1$:     MOV     #-1,R0          ;FILL R0 WITH ALL ONES
022230  013700  177574          MOV     SR1,R0          ;READ SR1 INTO R0
022234  001401                  BEQ     2$              ;BRANCH IF SR1 READS ALL ZEROS
022236  104014                  ERROR   14              ;SR1 DID NOT READ ALL ZEROS
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;000772
022240  012737  177777  172516  2$:     MOV     #-1,SR3         ;TRY TO WRITE ONES TO SR3
022246  022737  000060  172516          CMP     #60,SR3         ;ONLY BITS <5:4> SHOULD BE ONES
022254  001401                  BEQ     3$
022256  104012                  ERROR   12              ;DIDN'T READ BACK A "60"
022260  000005          3$:     RESET                   ;CLEARS SR3
022262  005737  172516          TST     SR3             ;VERIFY THAT IT WAS CLEARED
022266                  4$:
022266  001401                  BEQ     TST15           ;;BRANCH IF SR3 READ ALL ZEROS
022270  104012                  ERROR   12              ;SR3 DIDN'T READ ALL ZEROS


                        ;;*****************************************************************
                        ;*TEST 15       BIT TEST OF KERNEL & USER PAR'S
                        ;*
                        ;*      THE FOLLOWING TEST CHECKS THE BITS <15:00> OF BOTH THE KERNEL
                        ;*      AND USER PAGE ADDRESS REGISTERS.  A "0" IS ROTATED THRU
                        ;*      THE REGISTERS FROM LEFT TO RIGHT.
                        ;*
                        ;;*****************************************************************
022272  000004         TST15:  SCOPE
```

```
1787
1788  022274  012700  172340          1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR IN R0
1789  022300  012703  000010          2$:     MOV     #10,R3          ;SETUP R3 TO COUNT 8 PAR'S
1790  022304  012737  022312  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1791  022312  005010                  3$:     CLR     (R0)            ;CLEAR THE PAR
1792  022314  011001                          MOV     (R0),R1         ;READ THE PAR INTO R1
1793  022316  001401                          BEQ     4$              ;BRANCH IF PAR CLEARED OK
1794  022320  104011                          ERROR   11              ;PAR WOULD NOT CLEAR
1795                                                                  ;FOR TIGHTER SCOPE LOOP
1796                                                                  ;REPLACE ERROR CALL WITH
1797                                                                  ;"BR 3$" = 000774
1798  022322  012704  077777          4$:     MOV     #077777,R4      ;LOAD "WALKING 0" TEST PATTERN IN R4
1799  022326  012737  022334  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1800  022334  005010                  5$:     CLR     (R0)            ;CLEAR THE PAR BEFORE LOADING DATA
1801  022336  050410                          BIS     R4,(R0)         ;BIT SET THE TEST PATTERN INTO THE PAR
1802  022340  011002                          MOV     (R0),R2         ;READ THE PAR INTO R2
1803  022342  020402                          CMP     R4,R2           ;DOES DATA WRITTEN=DATA READ?
1804  022344  001402                          BEQ     6$              ;BRANCH IF YES
1805  022346  010401                          MOV     R4,R1           ;SETUP FOR ERROR REPORTING
1806  022350  104012                          ERROR   12              ;PAR BITS DID NOT SET CORRECTLY
1807                                                                  ;FOR TIGHTER SCOPE LOOP
1808                                                                  ;REPLACE ERROR CALL WITH
1809                                                                  ;"BR 5$" = 000767
1810  022352  000261                  6$:     SEC                     ;SET THE C-BIT FOR THE ROTATE INST.
1811  022354  006004                          ROR     R4              ;ROTATE THE TEST PATTERN IN R4
1812  022356  103766                          BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
1813  022360  062700  000002                 ADD     #2,R0           ;GET NEXT PAR ADDRESS IN R0
1814  022364  077326                          SOB     R3,3$           ;BRANCH BACK UNTIL ALL PAR'S TESTED
1815  022366  022700  177660                 CMP     #U1PAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
1816  022372  103003                          BHIS    7$              ;BRANCH IF YES
1817  022374  012700  177640                 MOV     #U1PAR0,R0      ;LOAD FIRST USER PAR ADDR. IN R0
1818  022400  000737                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
1819  022402  012737  022274  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP OR ERROR POINTER TO 1$
1820                                                                  ;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
```

```
1821                                    ;;*****************************************************************
1822                                    ;*TEST 16       BIT TEST OF KERNEL & USER PDR'S
1823                                    ;*
1824                                    ;*      THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
1825                                    ;*      KERNEL AND USER PAGE DESCRIPTOR REGISTERS.  A "0" IS ROTATED
1826                                    ;*      THRU THE REGISTERS FROM LEFT TO RIGHT.  SOME TEST PATTERNS WILL
1827                                    ;*      BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
1828                                    ;*
1829                                    ;;*****************************************************************
1830  022410  000004                    TST16:  SCOPE
1831
1832  022412  012700  172300            1$:     MOV     #K1PDR0,R0      ;LOAD ADDRESS OF FIRST PDR IN R0
1833  022416  012703  000010            2$:     MOV     #10,R3          ;SETUP R3 TO COUNT 8 PDR'S
1834  022422  012737  022430  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1835  022430  005010                    3$:     CLR     (R0)            ;CLEAR THE PDR
1836  022432  011001                            MOV     (R0),R1         ;READ THE PDR INTO R1
1837  022434  001401                            BEQ     4$              ;BRANCH IF PDR CLEARED OK
1838  022436  104011                            ERROR   11              ;PDR WOULD NOT CLEAR
1839                                                                    ;FOR TIGHTER SCOPE LOOP
1840                                                                    ;REPLACE ERROR CALL WITH
1841                                                                    ;"BR 3$" = 000774
1842  022440  012704  077777            4$:     MOV     #077777,R4      ;LOAD 'WALKING "0" TEST PATTERN IN R4
1843  022444  012737  022452  001110            MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1844  022452  005010                    5$:     CLR     (R0)            ;CLEAR THE PDR BEFORE LOADING DATA
1845  022454  010401                            MOV     R4,R1           ;LOAD DATA INTO R1
1846  022456  042701  100361                    BIC     #100361,R1      ;MASK UNUSED BITS OUT OF THE DATA
1847  022462  050110                            BIS     R1,(R0)         ;BIT SET THE TEST PATTERN INTO THE PDR
1848  022464  011002                            MOV     (R0),R2         ;READ THE PDR INTO R2
1849  022466  020102                            CMP     R1,R2           ;DOES DATA WRITTEN=DATA READ?
1850  022470  001401                            BEQ     6$              ;BRANCH IF YES
1851  022472  104012                            ERROR   12              ;PDR BITS DID NOT SET CORRECTLY
1852                                                                    ;FOR TIGHTER SCOPE LOOP
1853                                                                    ;REPLACE ERROR CALL WITH
1854                                                                    ;"BR 5$" = 000767
1855  022474  000261                    6$:     SEC                     ;SET THE C-BIT FOR THE ROTATE INST.
1856  022476  006004                            ROR     R4              ;ROTATE THE TEST PATTERN IN R4
1857  022500  103764                            BCS     5$              ;BRANCH BACK IF MORE BITS TO TEST
1858  022502  062700  000002                    ADD     #2,R0           ;GET NEXT PDR ADDRESS IN R0
1859  022506  077330                            SOB     R3,3$           ;BRANCH BACK UNTIL ALL PDR'S TESTED
1860  022510  022700  177620                    CMP     #U1PDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
1861  022514  103003                            BHIS    7$              ;BRANCH IF YES
1862  022516  012700  177600                    MOV     #U1PDR0,R0      ;LOAD FIRST USER PDR ADDR. IN R0
1863  022522  000735                            BR      2$              ;BRANCH BACK TO TEST USER PDR'S
1864  022524  012737  022412  001110    7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1865                                                                    ;LEAVE TEST WITH ALL WRITEABLE BITS IN
1866                                                                    ;ALL PDR'S = 1
1867
1868                                    ;;*****************************************************************
1869                                    ;*TEST 17       TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
1870                                    ;*
1871                                    ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
1872                                    ;*      PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
1873                                    ;*      THE OTHER.
1874                                    ;*
1875                                    ;;*****************************************************************
1876  022532  000004                    TST17:  SCOPE
```

```
 1877
 1878  022534  012700  172340          1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR INTO R0
 1879  022540  012737  022552  001110  2$.     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
 1880  022546  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER TO DO 8 PAR'S
 1881  022552  012701  177777          3$:     MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
 1882  022556  005010                          CLR     (R0)            ;CLEAR THE PAR
```

```
1883  022560  110110                          MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1884  022562  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1885  022564  042701  177400                  BIC     #177400,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1886  022570  020102                          CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO
1887  022572  001401                          BEQ     4$              ;BRANCH IF YES
1888  022574  104015                          ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1889                                                                  ;FOR TIGHTER SCOPE LOOP
1890                                                                  ;REPLACE ERROR CALL WITH
1891                                                                  ;"BR 3$" = 000766
1892  022576  012737  022604  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1893  022604  005010                  5$:     CLR     (R0)            ;CLEAR THE PAR
1894  022606  012701  177777                  MOV     #-1,R1          ;LOAD TEST, PATTERN INTO R1
1895  022612  110160  000001                  MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1896  022616  011002                          MOV     (R0),R2         ;READ THE ENTIRE PAR INTO R2
1897  022620  042701  000377                  BIC     #000377,R1      ;MASK LOW BYTE
1898  022624  020102                          CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1899  022626  001401                          BEQ     6$              ;BRANCH IF YES
1900  022630  104015                          ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1901                                                                  ;FOR TIGHTER SCOPE LOOP
1902                                                                  ;REPLACE ERROR CALL WITH
1903                                                                  ;"BR 5" = 000765
1904  022632  062700  000002          6$:     ADD     #2,R0           ;PUT ADDRESS OF NEXT PAR IN R0
1905  022636  077333                          SOB     R3,3$           ;BRANCH BACK UNTIL 8 PAR'S TESTED
1906  022640  022700  177660                  CMP     #U1PAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
1907  022644  103003                          BHIS    7$              ;BRANCH IF YES
1908  022646  012700  177640                  MOV     #U1PAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
1909  022652  000732                          BR      2$              ;BRANCH BACK TO TEST USER PAR'S
1910  022654  012737  022534  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1911
```

```
                                ;;*****************************************************************
                                ;*TEST 20       TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
                                ;*
                                ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
                                ;*      PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
                                ;*      THE OTHER.
                                ;*
                                ;;*****************************************************************
022662  000004                  TST20:  SCOPE

022664  012700  172300          1$:     MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST PDR INTO R0
022670  012737  022702  001110  2$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
022676  012703  000010                  MOV     #10,R3          ;LOAD LOOP COUNTER TO DO 8 PDR'S
022702  012701  177777          3$:     MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022706  005010                          CLR     (R0)            ;CLEAR THE PDR
022710  110110                          MOVB    R1,(R0)         ;WRITE 1'S TO THE LOW BYTE OF THE PDR
022712  011002                          MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022714  042701  177761                  BIC     #177761,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
022720  020102                          CMP     R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO?
022722  001401                          BEQ     4$              ;BRANCH IF YES
022724  104015                          ERROR   15              ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000766
022726  012737  022734  001110  4$:     MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
022734  005010                  5$:     CLR     (R0)            ;CLEAR THE PDR
022736  012701  177777                  MOV     #-1,R1          ;LOAD TEST PATTERN INTO R1
022742  110160  000001                  MOVB    R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
022746  011002                          MOV     (R0),R2         ;READ THE ENTIRE PDR INTO R2
022750  042701  100377                  BIC     #100377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
022754  020102                          CMP     R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
022756  001401                          BEQ     6$              ;BRANCH IF YES
022760  104015                          ERROR   15              ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000765
022762  062700  000002          6$:     ADD     #2,R0           ;PUT ADDRESS OF NEXT PDR IN R0
022766  077333                          SOB     R3,3$           ;BRANCH BACK UNTIL 8 PDR'S TESTED
022770  022700  177620                  CMP     #UIPDR7+2,R0    ;HAVE USER PDR'S BEEN TESTED?
022774  103003                          BHIS    7$              ;BRANCH IF YES
022776  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
023002  000732                          BR      2$              ;BRANCH BACK TO TEST USER PDR'S
023004  012737  022664  001110  7$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
```

```
                        ;;*****************************************************************
                        ;*TEST 21       PAR-PDR DUAL ADDRESSING TEST
                        ;*
                        ;*      THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
                        ;*      IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
                        ;*      SUBROUTINE AND THEN CLEARS JUST ONE OF THEM.  THE "CMPREG"
                        ;*      SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
                        ;*      THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
                        ;*      PAR OR PDR ADDRESS.  THE "CMPREG" SUBROUTINE REPORTS THE
                        ;*      ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
                        ;*      ANOTHER REGISTER WAS CLEARED.
                        ;;*****************************************************************
023012  000004          TST21:  SCOPE
023014  012737  023036  001110  1$:    MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER 2$
023022  012703  000010          MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023026  012700  172300          MOV     #K1PDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
023032  004737  035166          JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S IN PDR'S
023036  012706  001100          2$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023042  005010                  CLR     (R0)            ;CLEAR ONE OF THE KERNEL PDR'S
023044  004737  035260          JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023050  012720  177777          MOV     #-1,(R0)+       ;RESTORE ALL ONES, AND SETUP FOR NEXT PDR
023054  077310                  SOB     R3,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
023056  012737  023074  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
023064  012703  000010          MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023070  012700  172340          MOV     #K1PAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
023074  012706  001100          3$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023100  005010                  CLR     (R0)            ;CLEAR ONE OF THE KERNEL PAR'S
023102  004737  035260          JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023106  012720  177777          MOV     #-1,(R0)+       ;RESTORE ALL ONES, AND SETUP FOR NEXT PAR
023112  077310                  SOB     R3,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
023114  012737  023132  001110          MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
023122  012703  000010          MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023126  012700  177600          MOV     #U1PDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
023132  012706  001100          4$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023136  005010                  CLR     (R0)            ;CLEAR ONE OF THE USER PDR'S
023140  004737  035260          JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
023144  012720  177777          MOV     #-1,(R0)+       ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
023150  077310                  SOB     R3,4$           ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
023152  012737  023170  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
023160  012703  000010          MOV     #10,R3          ;LOAD LOOP COUNTER WITH AN 8
023164  012700  177640          MOV     #U1PAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
023170  012706  001100          5$:     MOV     #KERSTK,KSP     ;SETUP STACK POINTER
023174  005010                  CLR     (R0)            ;CLEAR ONE OF THE USER PAR'S
023176  004737  035260          JSR     PC,CMPREG       ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
```

```
2005  023202  012720  177777          MOV     #-1,(R0)+       ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
2006  023206  077310                  SOB     R3,5$           ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
2007  023210  012737  023014  001110  MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2008
2009                                  ;;*****************************************************************
2010                                  ;*TEST 22       TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
2011                                  ;*
2012                                  ;*      THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
2013                                  ;*      NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION.  THE
2014                                  ;*      "SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
2015                                  ;*      THE PAR/PDR'S.  THEN THEY ARE READ TO SEE THAT THEY REMAINED
2016                                  ;*      UNCHANGED
2017                                  ;*
2018                                  ;;*****************************************************************
2019  023216  000004          TST22:  SCOPE
2020
2021
2022  023220  004737  035166  1$:     JSR     PC,SETREG       ;SET ALL BITS IN ALL PAR'S AND PDR'S
2023  023224  000005                  RESET                   ;ISSUE AN "INIT" BY EXECUTING A RESET
2024  023226  012700  172300          MOV     #K1PDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2025  023232  012704  000010          MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2026  023236  011001          2$:     MOV     (R0),R1         ;READ A KERNEL PDR INTO R1
2027  023240  022701  077416          CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
2028  023244  001401                  BEQ     3$              ;BRANCH IF YES
2029  023246  104055                  ERROR   55              ;KERNEL PDR AFFECTED BY A RESET
2030                                                          ;FOR TIGHTER SCOPE LOOP
2031                                                          ;REPLACE ERROR CALL WITH
2032                                                          ;"BR 2$" = 000773
2033  023250  062700  000002  3$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR
2034  023254  077410                  SOB     R4,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2035  023256  012700  172340          MOV     #K1PAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2036  023262  012704  000010          MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2037  023266  011001          4$:     MOV     (R0),R1         ;READ A KERNEL PAR INTO R1
2038  023270  022701  177777          CMP     #177777,R1      ;ARE ALL THE BITS STILL SET?
2039  023274  001401                  BEQ     5$              ;BRANCH IF YES
2040  023276  104055                  ERROR   55              ;KERNEL PAR AFFECTED BY A RESET
2041                                                          ;FOR TIGHTER SCOPE LOOP
2042                                                          ;REPLACE ERROR CALL WITH
2043                                                          ;"BR 4$" = 000773
2044  023300  062700  000002  5$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR
2045  023304  077410                  SOB     R4,4$           ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2046  023306  012700  177600          MOV     #U1PDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2047  023312  012704  000010          MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2048  023316  011001          6$:     MOV     (R0),R1         ;READ A USER PDR INTO R1
2049  023320  022701  077416          CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
2050  023324  001401                  BEQ     7$              ;BRANCH IF YES
2051  023326  104055                  ERROR   55              ;USER PDR AFFECTED BY A RESET
2052                                                          ;FOR TIGHTER SCOPE LOOP
2053                                                          ;REPLACE ERROR CALL WITH
2054                                                          ;"BR 6$" = 000773
2055  023330  062700  000002  7$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PDR
2056  023334  077410                  SOB     R4,6$           ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
2057
2058  023336  012700  177640          MOV     #U1PAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2059  023342  012704  000010          MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
2060  023346  011001          8$:     MOV     (R0),R1         ;READ A USER PAR INTO R1
```

```
023350  022701  177777          CMP     #177777,R1      ;ARE ALL THE BITS STILL SET?
023354  001401                  BEQ     9$              ;BRANCH IF YES
023356  104055                  ERROR   55              ;USER PAR AFFECTED BY A RESET .
                                                        ;FOR TIGHTER SCOPE LOOP
                                                        ;REPLACE ERROR CALL WITH
                                                        ;"BR 8$" = 000773
023360  062700  000002  9$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT USER PAR
023364  077410                  SOB     R4,8$           ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
```

```
;;*****************************************************************
;*TEST 23       RELOCATION & ADDER TEST (NO CARRIES)
;*
;*      THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
;*      FOR THE REST OF THE PROGRAM.  IT THEN USES DIFFERENT
;*      VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
;*      TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE
;*      MEMORY MANAGEMENT ADDER.  THE VALUES ARE SUCH
;*      THAT NO CARRIES ARE GENERATED OUT OF THE ADDER.
;*
;*      THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
;*      IS FORMED BY THE ADDER IS TO WRITE A PATTERN TO VIRTUAL
;*      LOCATION WITH MEMORY MGMT., AND
;*      THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
;*      HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
;*      22-BIT AND 18-BIT ADDRESSING ARE USED.
;;*****************************************************************
023366  000004          TST23:  SCOPE

023370  012700  172340  1$:     MOV     #KIPAR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
023374  005001                  CLR     R1              ;CLEAR R1
023376  012702  000007          MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
023402  010120          2$:     MOV     R1,(R0)+        ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
023404  062701  000200          ADD     #200,R1
023410  077204                  SOB     R2,2$           ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
```

```
2117  023412  012710  007600                  MOV     #7600,(R0)      ;MAP KIPAR7 TO THE I/O PAGE
2118  023416  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2119  023422  012701  077406                  MOV     #77406,R1       ;LOAD PDR DATA INTO R1
2120  023426  012702  000010                  MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
2121  023432  010120                  3$:     MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2122  023434  077202                          SOB     R2,3$           ;       EXPANDABLE, READ/WRITE
2123
2124  023436  012737  023436  001110  4$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2125  023444  012700  067776                  MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2126  023450  012701  107776                  MOV     #107776,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2127  023454  012702  125250                  MOV     #125250,R2      ;LOAD TEST PATTERN INTO R2
2128  023460  012704  000600                  MOV     #600,R4 ;LOAD R4 WITH PAR VALUE
2129  023464  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2130  023470  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2131  023474  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
2132  023502  010211                          MOV     R2,(R1)         ;LOAD 125250 USING ADDER  (PAR4 + VIRT ADDR.)
2133  023504  000005                          RESET                   ;TURN OFF MEMORY MGMT.
2134  023506  011003                          MOV     (R0),R3         ;READ 125250 BACK WITHOUT USING MEM. MGMT.
2135  023510  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2136  023514  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2137                                                                  ;WRITTEN USING MEMORY MANAGEMENT?
2138  023516  001405                          BEQ     5$              ;BRANCH IF YES
2139  023520  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2140  023524  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2141  023530  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2142                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2143                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2144                                                                  ;FORMED WRONG BY ADDERS USING
2145                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2146                                                                  ;FOR TIGHTER SCOPE LOOP
2147                                                                  ;REPLACE ERROR CALL WITH
2148                                                                  ;"BR 4$" = 000742
2149  023532                          5$:
2150  023532  012737  023532  001110  6$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2151  023540  012700  067776                  MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2152  023544  012701  102576                  MOV     #102576,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2153  023550  012702  125251                  MOV     #125251,R2      ;LOAD TEST PATTERN INTO R2
2154  023554  012704  000652                  MOV     #652,R4 ;LOAD R4 WITH PAR VALUE
2155  023560  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2156  023564  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2157  023570  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
2158  023576  010211                          MOV     R2,(R1)         ;LOAD 125251 USING ADDER  (PAR4 + VIRT ADDR.)
2159  023600  000005                          RESET                   ;TURN OFF MEMORY MGMT.
2160  023602  011003                          MOV     (R0),R3         ;READ 125251 BACK WITHOUT USING MEM. MGMT.
2161  023604  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2162  023610  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2163                                                                  ;WRITTEN USING MEMORY MANAGEMENT?
2164  023612  001405                          BEQ     7$              ;BRANCH IF YES
2165  023614  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2166  023620  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2167  023624  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2168                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2169                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2170                                                                  ;FORMED WRONG BY ADDERS USING
2171                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2172                                                                  ;FOR TIGHTER SCOPE LOOP
```

```
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 6$" = 000742
023626                                  7$:
023626  012737  023626  001110  8$:     MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
023634  012700  067776                  MOV     #67776,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
023640  012701  105276                  MOV     #105276,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
023644  012702  125252                  MOV     #125252,R2      ;LOAD TEST PATTERN INTO R2
023650  012704  000625                  MOV     #625,R4 ;LOAD R4 WITH PAR VALUE
023654  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
023660  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
023664  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
023672  010211                          MOV     R2,(R1)         ;LOAD 125252 USING ADDER  (PAR4 + VIRT ADDR.)
023674  000005                          RESET                   ;TURN OFF MEMORY MGMT.
023676  011003                          MOV     (R0),R3         ;READ 125252 BACK WITHOUT USING MEM. MGMT.
023700  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
023704  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
                                                                ;WRITTEN USING MEMORY MANAGEMENT?
023706  001405                          BEQ     9$              ;BRANCH IF YES
023710  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
023714  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
023720  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$" = 000742
023722                                  9$:
                                                                ;SET LOOP ON ERROR POINTER TO 10$
023722  012737  023722  001110  10$:    MOV     #10$,$LPERR     ;LOAD PHYS. ADDR. OF PSW INTO R0
023730  012700  177776                  MOV     #PSW,R0         ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
023734  012701  100076                  MOV     #100076,R1      ;LOAD DATA FOR PSW IN R2
023740  012702  030340                  MOV     #030340,R2      ;LOAD R4 WITH PAR VALUE
023744  012704  007777                  MOV     #7777,R4        ;LOAD KERNEL PAR 4 BITS <11:00>
023750  010437  172350                  MOV     R4,KIPAR4       ;CLEAR THE PSW
023754  005010                          CLR     (R0)            ;TURN ON "MEMORY MANAGEMENT" (18 BIT ADDRESSING)
023756  052737  000001  177572          BIS     #BIT0,SR0       ;LOAD PSW USING ADDER (PAR4 + VIRT ADDR.)
023764  010211                          MOV     R2,(R1)         ;TURN OFF MEM. MGMT (SR0=0)
023766  000005                          RESET                   ;READ PSW BACK WITHOUT USING MEM. MGMT.
023770  011003                          MOV     (R0),R3         ;CLEAR THE PSW
023772  005010                          CLR     (R0)            ;MASK T-BIT & CC BITS OUT OF DATA READ
023774  042703  000037                  BIC     #37,R3          ;WAS PSW WRITTEN?
024000  020203                          CMP     R2,R3           ;BRANCH IF YES
024002  001405                          BEQ     11$             ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
024004  010137  001306                  MOV     R1,VIRT1        ;GO FORM PHYSICAL ADDR. FOR TYPING
024010  004737  035452                  JSR     PC,FORMPA       ;PSW DID NOT HAVE DATA THAT IT SHOULD HAVE.
024014  104017                          ERROR   17              ;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$" = 000742
024016  012737  024016  001110  11$:    MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
024024  012700  177776                  MOV     #PSW,R0         ;LOAD PHYS. ADDR. OF PSW INTO R0
```

```
2229  024030  012701  117776                  MOV     #117776,R1      ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2230  024034  012702  030240                  MOV     #030240,R2      ;LOAD DATA FOR PSW IN R2
2231  024040  012704  177600                  MOV     #177600,R4      ;LOAD R4 WITH PAR VALUE
2232  024044  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2233  024050  052737  000021  177572          BIS     #21,SR0         ;TURN ON 'MEMORY MANAGEMENT''(22 BIT ADDRESSING)
2234  024056  010211                          MOV     R2,(R1)         ;LOAD PSW USING ADDER (PAR4 + VIRT. ADDR.)
2235  024060  000005                          RESET                   ;TURN OFF MEM. MGMT (SR0=0)
2236  024062  011003                          MOV     (R0),R3         ;READ PSW BACK WITHOUT USING MEM. MGMT.
2237  024064  005010                          CLR     (R0)            ;CLEAR THE PSW
2238  024066  042703  000037                  BIC     #37,R3          ;MASK T-BIT & CC BITS OUT OF DATA READ
2239  024072  020203                          CMP     R2,R3           ;WAS PSW WRITTEN?
2240  024074  001405                          BEQ     12$             ;BRANCH IF YES
2241  024076  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
2242  024102  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDR. FOR TYPING
2243  024106  104017                          ERROR   17              ;PSW DID NOT HAVE DATA THAT IT SHOULD
2244                                                                  ;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2245                                                                  ;NOT FORMED BY ADDERS USING THE
2246                                                                  ;VIRTUAL ADDR. AND KIPAR4
2247                                                                  ;FOR TIGHTER SCOPE LOOP
2248                                                                  ;REPLACE ERROR CALL WITH
2249                                                                  ;"BR 11$" = 000743
2250  024110  012737  023370  001110  12$:    MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2251
2252                          ;;*****************************************************************
2253                          ;*TEST 24       RELOCATION & ADDER TEST (WITH CARRIES)
2254                          ;*
2255                          ;*      THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2256                          ;*      TEST TO VERIFY MEMORY MANAGEMENTS ABILITY TO CONSTRUCT
2257                          ;*      PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2258                          ;*      CONTENTS OF A PAGE ADDRESS REGISTER.  HOWEVER, THE VALUES
2259                          ;*      AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES
2260                          ;*      AND CHECK "WRAPAROUND" TO ADDRESS 000000 BY
2261                          ;*      USING VIRTUAL ADDR. 111400 AND KIPAR4 = 177664.
2262                          ;*      22-BIT ADDRESSING IS USED.
2263                          ;;*****************************************************************
2264  024116  000004          TST24:  SCOPE
2265
2266  024120                  1$:                                     ;KERNEL PAR'S AND PDR'S HAVE BEEN
2267                                                                  ;SETUP BY THE PREVIOUS TEST
2268  024120  012737  024120  001110  2$:      MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2269  024126  012700  066476                  MOV     #66476,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2270  024132  012701  114376                  MOV     #114376,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2271  024136  012702  125253                  MOV     #125253,R2      ;LOAD TEST PATTERN INTO R2
2272  024142  012704  000521                  MOV     #521,R4 ;LOAD R4 WITH PAR VALUE
2273  024146  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2274  024152  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2275  024156  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
2276  024164  010211                          MOV     R2,(R1)         ;LOAD 125253 USING ADDER  (PAR4 + VIRT ADDR.)
2277  024166  000005                          RESET                   ;TURN OFF MEMORY MGMT.
2278  024170  011003                          MOV     (R0),R3         ;READ 125253 BACK WITHOUT USING MEM. MGMT.
2279  024172  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2280  024176  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2281                                                                  ;WRITTEN USING MEMORY MANAGEMENT?
2282  024200  001405                          BEQ     3$              ;BRANCH IF YES
2283  024202  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2284  024206  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
```

```
2285  024212  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2286                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2287                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2288                                                                  ;FORMED WRONG BY ADDERS USING
2289                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2290                                                                  ;FOR TIGHTER SCOPE LOOP
2291                                                                  ;REPLACE ERROR CALL WITH
2292                                                                  ;"BR 2$" = 000742
2293  024214                          3$:
2294  024214  012737  024214  001110  4$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2295  024222  012700  062276                  MOV     #62276,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2296  024226  012701  107376                  MOV     #107376,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2297  024232  012702  125254                  MOV     #125254,R2      ;LOAD TEST PATTERN INTO R2
2298  024236  012704  000527                  MOV     #527,R4 ;LOAD R4 WITH PAR VALUE
2299  024242  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2300  024246  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2301  024252  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
2302  024260  010211                          MOV     R2,(R1)         ;LOAD 125254 USING ADDER  (PAR4 + VIRT ADDR.)
2303  024262  000005                          RESET                   ;TURN OFF MEMORY MGMT.
2304  024264  011003                          MOV     (R0),R3         ;READ 125254 BACK WITHOUT USING MEM. MGMT.
2305  024266  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2306  024272  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2307                                                                  ;WRITTEN USING MEMORY MANAGEMENT?
2308  024274  001405                          BEQ     5$              ;BRANCH IF YES
2309  024276  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2310  024302  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2311  024306  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2312                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2313                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2314                                                                  ;FORMED WRONG BY ADDERS USING
2315                                                                  ;THE VIRTUAL ADDR. AND KIPAR4
2316                                                                  ;FOR TIGHTER SCOPE LOOP
2317                                                                  ;REPLACE ERROR CALL WITH
2318                                                                  ;"BR 4$" = 000742
2319  024310                          5$:
2320  024310  012737  024310  001110  6$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2321  024316  012700  062076                  MOV     #62076,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
2322  024322  012701  104576                  MOV     #104576,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
2323  024326  012702  125255                  MOV     #125255,R2      ;LOAD TEST PATTERN INTO R2
2324  024332  012704  000553                  MOV     #553,R4 ;LOAD R4 WITH PAR VALUE
2325  024336  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
2326  024342  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
2327  024346  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
2328  024354  010211                          MOV     R2,(R1)         ;LOAD 125255 USING ADDER  (PAR4 + VIRT ADDR.)
2329  024356  000005                          RESET                   ;TURN OFF MEMORY MGMT.
2330  024360  011003                          MOV     (R0),R3         ;READ 125255 BACK WITHOUT USING MEM. MGMT.
2331  024362  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2332  024366  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
2333                                                                  ;WRITTEN USING MEMORY MANAGEMENT?
2334  024370  001405                          BEQ     7$              ;BRANCH IF YES
2335  024372  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2336  024376  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
2337  024402  104017                          ERROR   17              ;TEST LOCATION DID NOT HAVE PATTERN
2338                                                                  ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2339                                                                  ;APPARENTLY PHYSICAL ADDR. WAS
2340                                                                  ;FORMED WRONG BY ADDERS USING
```

```
                                                                ;THE VIRTUAL ADDR. AND KIPAR4
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 6$" = 000742
024404                                          7$:
024404  012737  024404  001110  8$:     MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
024412  012700  000000                  MOV     #00000,R0       ;LOAD PHYSICAL ADDR. PBA INTO R0
024416  012701  111400                  MOV     #111400,R1      ;LOAD VIRTUAL ADDR. VBA INTO R1
024422  012702  125256                  MOV     #125256,R2      ;LOAD TEST PATTERN INTO R2
024426  012704  177664                  MOV     #177664,R4      ;LOAD R4 WITH PAR VALUE
024432  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR 4 BITS <15:00>
024436  011037  001176                  MOV     (R0),$TMP0      ;SAVE CONTENTS AT TEST LOCATION
024442  052737  000021  177572          BIS     #21,SR0         ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
024450  010211                          MOV     R2,(R1)         ;LOAD 125256 USING ADDER  (PAR4 + VIRT ADDR.)
024452  000005                          RESET                   ;TURN OFF MEMORY MGMT.
024454  011003                          MOV     (R0),R3         ;READ 125256 BACK WITHOUT USING MEM. MGMT.
024456  013710  001176                  MOV     $TMP0,(R0)      ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
024462  020203                          CMP     R2,R3           ;WAS SAME PATTERN READ BACK THAT WAS
                                                                ;WRITTEN USING MEMORY MANAGEMENT?
024464  001405                          BEQ     9$              ;BRANCH IF YES
024466  010137  001306                  MOV     R1,VIRT1        ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
024472  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS FOR TYPING
024476  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$" = 000742
024500                                  9$:
024500  012737  024120  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                        ;;*****************************************************************
                        ;*TEST 25       READ AND WRITE WHILE IN RELOCATE MODE
                        ;*
                        ;*      THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
                        ;*      READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
                        ;*      060000-067600.  ONE LOCATION IN EVERY BLOCK (32. WORDS)
                        ;*      IS WRITTEN USING PAR4 AND READ USING PAR5.  THIS IS
                        ;*      DONE IN BOTH USER AND KERNEL MODES.  THE USER PAR/PDR'S
                        ;*      ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
                        ;*      MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
                        ;*      OF THE PROGRAM.  THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
                        ;*      IS CHECKED BY READING AND WRITING IN USER MODE.  REMEMBER
                        ;*      ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
                        ;*      MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
                        ;*      THE PAR/PDR'S TO EXECUTE.
                        ;*
                        ;*      WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
                        ;*      NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
                        ;*      PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
                        ;*      USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
                        ;*      USER MODE).  IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
                        ;*      TO 9$ WHERE THE TRAP IS REPORTED.
                        ;*
```

```
                                        ;*      BY SETTING THE LOCATION $MADR1 IN THE E-TABLE TO A CONSTANT,
                                        ;*      AS DESCRIBED IN THE DOCUMENTATION, THIS TEST WILL CONTINUE
                                        ;*      ACCESSING LOCATIONS THROUGHOUT MEMORY BY INCREASING THE VALUE
                                        ;*      OF PAR4 AND PAR5.
                                        ;;*****************************************************************
024506  000004                  TST25:  SCOPE

024510  005037  177776          1$:     CLR     PSW             ;START IN KERNEL MODE
024514  012704  000577                  MOV     #577,R4         ;LOAD R4 WITH VALUE FOR PAR4
024520  012705  000600                  MOV     #600,R5         ;LOAD R5 WITH VALUE FOR PAR5
024524  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNEL PAR4
024530  010537  172352                  MOV     R5,KIPAR5       ;LOAD KERNEL PAR5
024534  012700  177640                  MOV     #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
024540  005001                          CLR     R1              ;CLEAR R1
024542  012702  000007                  MOV     #7,R2           ;LOAD LOOP COUNTER WITH A 7
024546  010120                  2$:     MOV     R1,(R0)+        ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
024550  062701  000200                  ADD     #200,R1
024554  077204                          SOB     R2,2$           ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
024556  012710  177600                  MOV     #177600,(R0)    ;MAP USER PAR7 TO THE I/O PAGE
024562  012700  177600                  MOV     #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
024566  012701  077406                  MOV     #77406,R1       ;LOAD PDR DATA INTO R1
024572  012702  000010                  MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
024576  010120                  3$:     MOV     R1,(R0)+        ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
024600  077202                          SOB     R2,3$           ;       EXPANDABLE, READ/WRITE
024602  012737  024662  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
024610  012737  025146  000250          MOV     #9$,MMVEC       ;SET M. M. TRAP VECTOR TO 9$
024616  012737  000021  177572          MOV     #21,SR0         ;TURN ON MEMORY MANAGEMENT(22-BIT ADDRESSING)
024624  105037  177610          10$:    CLRB    UIPDR4          ;MAP USER SPACE NON-RESIDENT WHILE
024630  105037  177612                  CLRB    UIPDR5          ;       TESTING KERNEL SPACE
024634  010537  177650                  MOV     R5,UIPAR4       ;MAP USER PAR'S OPPOSITE OF KIPAR'S
024640  010437  177652                  MOV     R4,UIPAR5
024644  013737  177776  001176  4$:     MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
024652  012700  100100                  MOV     #100100,R0      ;PUT VIRTUAL ADDR. THAT USER PAR4 IN R0
024656  012701  120000                  MOV     #120000,R1      ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
024662  010010                  5$:     MOV     R0,(R0)         ;WRITE TO TEST LOC. USING PAR4
024664  011102                          MOV     (R1),R2         ;READ THE SAME LOC., BUT USING PAR5
024666  020002                          CMP     R0,R2           ;DID WE READ WHAT WE WROTE?
024670  001411                          BEQ     6$              ;BRANCH IF YES
024672  010137  001310                  MOV     R1,VIRT2        ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
024676  010037  001306                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
024702  004737  035452                  JSR     PC,FORMPA       ;GO FORM PHYSICAL ADDRESS  BEING USED
024706  104020                          ERROR   20              ;READING LOC. USING PAR5 AND A VIRT.
                                                                ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
                                                                ;PAR4 AND VIRT. ADDRESS.
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000765
024710  013700  001306                  MOV     VIRT1,R0        ;RESTOPE VBA IN R0
024714  062700  000100          6$:     ADD     #100,R0         ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
024720  062701  000100                  ADD     #100,R1
024724  020127  127700                  CMP     R1,#127700      ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
024730  001354                          BNE     5$              ;BRANCH IF NO
024732  032737  140000  177776          BIT     #140000,PSW     ;HAVE WE DONE TEST IN USER MODE YET?
024740  001026                          BNE     7$              ;BRANCH IF YES
024742  010437  177650                  MOV     R4,UIPAR4       ;LOAD USER PAR4
024746  010537  177652                  MOV     R5,UIPAR5       ;LOAD USER PAR5
```

```
 2453  024752  112737  000006  177610          MOVB    #6,UIPDR4       ;MAP USER SPACE R/W TO TEST IT
 2454  024760  112737  000006  177612          MOVB    #6,UIPDR5
 2455  024766  105037  172310                  CLRB    KIPDR4          ;MAP  KERNEL SPACE NON-RESIDENT WHILE
 2456  024772  105037  172312                  CLRB    KIPDR5          ;       TESTING USER SPACE
 2457  024776  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
 2458  025002  010437  172352                  MOV     R4,KIPAR5
 2459  025006  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
 2460  025014  000713                          BR      4$              ;GO BACK AND READ/WRITE IN USER MODE
 2461  025016  005037  177776          7$:     CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
 2462
 2463  025022  020537  001260                  CMP     R5,@#$MADR1     ;HAVE WE CHECKED ALL MEMORY?
 2464  025026  002020                          BGE     8$              ;BRANCH IF YES
 2465  025030  062704  000200                  ADD     #200,R4         ;LOAD WITH NEXT VALUE FOR PAR4
 2466  025034  062705  000200                  ADD     #200,R5         ;LOAD WITH NEXT VALUE FOR PAR5
 2467  025040  010437  172350                  MOV     R4,KIPAR4       ;LOAD KERNAL PAR4
 2468  025044  010537  172352                  MOV     R5,KIPAR5       ;LOAD KERNAL PAR5
 2469  025050  112737  000006  172310          MOVB    #6,KIPDR4       ;MAP KERNAL SPACE R/W WHILE TESTING
 2470  025056  112737  000006  172312          MOVB    #6,KIPDR5
 2471  025064  000137  024624                  JMP     10$             ;CONTINUE TEST
 2472
 2473  025070  012737  077406  172310  8$:     MOV     #77406,KIPDR4   ;REMAP KERNEL PAGES READ/WRITE
 2474  025076  012737  077406  172312          MOV     #77406,KIPDR5
 2475  025104  012705  000600                  MOV     #600,R5
 2476  025110  010537  172350                  MOV     R5,KIPAR4       ;MAP KERNEL  AND USER PAR'S 4 & 5
 2477  025114  010537  172352                  MOV     R5,KIPAR5       ;  BACK TO 12-16K
 2478  025120  010537  177650                  MOV     R5,UIPAR4
 2479  025124  010537  177652                  MOV     R5,UIPAR5
 2480  025130  012737  002150  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
 2481  025136  012737  024510  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
 2482  025144  000427                          BR      TST26           ;;GO TO NEXT TEST
 2483
 2484  025146  012637  001266          9$:     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
 2485  025152  012637  001270                  MOV     (KSP)+,TRAPPS
 2486                                                                  ;PROGRAM WILL TRAP TO HERE IF TRY
 2487                                                                  ;TO USE USER PDR'S WHEN IN KERNEL MODE
 2488                                                                  ;OR KERNEL PDR'S WHEN IN USER MODE
 2489  025156  010037  001306                  MOV     R0,VIRT1        ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
 2490  025162  004737  035452                  JSR     PC,FORMPA       ;GO FORM THE PHYSICAL ADDRESS  BEING USED
 2491  025166  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE SR0 & SR2  FOR ERROR REPORT
 2492  025174  013737  177576  001274          MOV     SR2,WASSR2
 2493  025202  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
 2494  025210  104052                          ERROR   52              ;M.M. TRAP WHILE IN RELOCATE MODE -
 2495                                                                  ;REFERENCED WRONG SET OF PDR'S
 2496                                                                  ;FOR TIGHTER SCOPE LOOP
 2497                                                                  ;REPLACE ERROR CALL WITH
 2498                                                                  ;A "NOP" = 000240
 2499  025212  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
 2500  025216  013746  001266                  MOV     TRAPPC,-(KSP)
 2501  025222  000002                          RTI                     ;RETURN TO TEST
 2502
 2503
 2504                                  ;;*****************************************************************
 2505                                  ;*TEST 26       W-BIT LOGIC TEST, KERNEL PDR'S
 2506                                  ;*
 2507                                  ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
 2508                                  ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
```

```
                                ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
                                ;*      77776,77776, & 777776 RESPECTIVELY).
                                ;*      WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
                                ;*      EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS.  THE PDR'S
                                ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
                                ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
                                ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  KERNEL PDR'S 3,4,5,6
                                ;*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
                                ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO.  THE
                                ;*      W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
                                ;;*****************************************************************
025224  000004                  TST26:  SCOPE
025226                          1$:
025226  004737  035100                  JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
025232  012702  000004                  MOV     #4,R2           ;SET LOOP COUNTER TO 4
025236  012700  172346                  MOV     #KIPAR3,R0      ;LOAD ADDRESS OF PAR3 INTO R0
025242  012701  000600                  MOV     #600,R1         ;LOAD "12-16K" PAR VALUE INTO R1
025246  010120                  2$:     MOV     R1,(R0)+        ;MAP PARS 3-6 TO 12-16K
025250  077202                          SOB     R2,2$           ;LOOP TIL ALL 4 OF THEM LOADED
025252  012705  172300                  MOV     #KIPDR0,R5      ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
025256  012704  000010                  MOV     #10,R4          ;SET LOOP COUNTER TO 8
025262  012703  017776                  MOV     #17776,R3       ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
025266  012737  025274  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
025274  012700  172300          3$:     MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
025300  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
025304  012701  077406                  MOV     #77406,R1       ;PUT "W-BIT OFF DATA" INTO R1
025310  010120                  4$:     MOV     R1,(R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
025312  077202                          SOB     R2,4$           ;LOOP UNTIL ALL OF THEM SETUP
025314  011313                          MOV     (R3),(R3)       ;DO "DATO" TO VIRTUAL ADDR.-SETTING A W-BIT
025316  031527  000100                  BIT     (R5),#WBIT      ;DID THAT CAUSE W-BIT TO BE SET?
025322  001002                          BNE     5$              ;BRANCH IF YES
025324  104021                          ERROR   21              ;W-BIT DID NOT GET SET IN PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000763
025326  000422                          BR      8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
025330  012702  000010          5$:     MOV     #10,R2          ;SET LOOP COUNTER TO 8
025334  012700  172300                  MOV     #KIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
025340  031027  000100          6$:     BIT     (R0),#WBIT      ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
025344  001403                          BEQ     7$              ;BRANCH IF YES
025346  020500                          CMP     R5,R0           ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
025350  001401                          BEQ     7$              ;BRANCH IF YES
025352  104022                          ERROR   22              ;W-BIT GOT SET IN MORE THAN ONE PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000750
025354  062700  000002          7$:     ADD     #2,R0           ;POINT R0 TO NEXT PDR TO BE CHECKED
025360  077211                          SOB     R2,6$           ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
025362  010115                          MOV     R1,(R5)         ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
025364  031527  000100                  BIT     (R5),#WBIT      ;DID WRITING PDR CLEAR THE W-BIT?
025370  001401                          BEQ     8$              ;BRANCH IF YES
025372  104023                          ERROR   23              ;W-BIT DID NOT CLEAR BY WRITING THE PDR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 3$" = 000740
025374  062705  000002          8$:     ADD     #2,R5           ;POINT R5 TO THE NEXT PDR TO BE TESTED
```

```
2565  025400  062703  020000                ADD     #20000,R3       ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2566  025404  077445                        SOB     R4,3$           ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2567  025406  012737  025226  001110        MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2568  025414  004737  035134                JSR     PC,TON          ;TURN T-BIT BACK ON FOR NEXT TEST
2569
2570                                  ;;*****************************************************************
2571                                  ;*TEST 27       W-BIT LOGIC TEST, USER PDR'S
2572                                  ;*
2573                                  ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2574                                  ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2575                                  ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2576                                  ;*      77776,77776, & 777776 RESPECTIVELY).
2577                                  ;*      WHICH SHOULD CAUSE THE 'W-BIT" TO SET IN EACH OF THE
2578                                  ;*      EIGHT (8) USER PAGE DESCRIPTOR REGISTERS.  THE PDR'S
2579                                  ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2580                                  ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2581                                  ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  USER PDR'S 3,4,5,6
2582                                  ;*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
2583                                  ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO.  THE
2584                                  ;*      W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
2585                                  ;;*****************************************************************
2586  025420  000004                TST27:  SCOPE
2587  025422  012737  140000  177776 1$:     MOV     #140000,PSW     ;GO TO USER MODE FOR THIS TEST
2588  025430  004737  035100                JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2589  025434  012702  000004                MOV     #4,R2           ;SET LOOP COUNTER TO 4
2590  025440  012700  177646                MOV     #UIPAR3,R0      ;LOAD ADDRESS OF PAR3 INTO R0
2591  025444  012701  000600                MOV     #600,R1         ;LOAD "12-16K" PAR VALUE INTO R1
2592  025450  010120                2$:     MOV     R1,(R0)+        ;MAP PARS 3-6 TO 12-16K
2593  025452  077202                        SOB     R2,2$           ;LOOP TIL ALL 4 OF THEM LOADED
2594  025454  012705  177600                MOV     #UIPDR0,R5      ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2595  025460  012704  000010                MOV     #10,R4          ;SET LOOP COUNTER TO 8
2596  025464  012703  017776                MOV     #17776,R3       ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2597  025470  012737  025476  001110        MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2598  025476  012700  177600        3$:     MOV     #UIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2599  025502  012702  000010                MOV     #10,R2          ;SET LOOP COUNTER TO 8
2600  025506  012701  077406                MOV     #77406,R1       ;PUT 'W-BIT OFF DATA" INTO R1
2601  025512  010120                4$:     MOV     R1,(R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2602  025514  077202                        SOB     R2,4$           ;LOOP UNTIL ALL OF THEM SETUP
2603  025516  011313                        MOV     (R3),(R3)       ;DO "DATO" TO VIRTUAL ADDR.-SETTING A W-BIT
2604  025520  031527  000100                BIT     (R5),#WBIT      ;DID THAT CAUSE W-BIT TO BE SET?
2605  025524  001002                        BNE     5$              ;BRANCH IF YES
2606  025526  104021                        ERROR   21              ;W-BIT DID NOT GET SET IN PDR
2607                                                                ;FOR TIGHTER SCOPE LOOP
2608                                                                ;REPLACE ERROR CALL WITH
2609                                                                ;"BR 3$" = 000763
2610  025530  000422                        BR      8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2611  025532  012702  000010        5$:     MOV     #10,R2          ;SET LOOP COUNTER TO 8
2612  025536  012700  177600                MOV     #UIPDR0,R0      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2613  025542  031027  000100        6$:     BIT     (R0),#WBIT      ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2614  025546  001403                        BEQ     7$              ;BRANCH IF YES
2615  025550  020500                        CMP     R5,R0           ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2616  025552  001401                        BEQ     7$              ;BRANCH IF YES
2617  025554  104022                        ERROR   22              ;W-BIT GOT SET IN MORE THAN ONE PDR
2618                                                                ;FOR TIGHTER SCOPE LOOP
2619                                                                ;REPLACE ERROR CALL WITH
2620                                                                ;"BR 3$" = 000750
```

```
2621  025556  062700  000002          7$:     ADD     #2,R0           ;POINT R0 TO NEXT PDR TO BE CHECKED
2622  025562  077211                          SOB     R2,6$           ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2623  025564  010115                          MOV     R1,(R5)         ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2624  025566  031527  000100                  BIT     (R5),#WBIT      ;DID WRITING PDR CLEAR THE W-BIT?
2625  025572  001401                          BEQ     8$              ;BRANCH IF YES
2626  025574  104023                          ERROR   23              ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2627                                                                  ;FOR TIGHTER SCOPE LOOP
2628                                                                  ;REPLACE ERROR CALL WITH
2629                                                                  ;"BR 3$" = 000740
2630  025576  062705  000002          8$:     ADD     #2,R5           ;POINT R5 TO THE NEXT PDR TO BE TESTED
2631  025602  062703  020000                  ADD     #20000,R3       ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2632  025606  077445                          SOB     R4,3$           ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2633  025610  012737  025422  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2634  025616  004737  035134                  JSR     PC,TON          ;TURN T-BIT BACK ON FOR NEXT TEST
2635  025622  005037  177776                  CLR     PSW             ;BACK TO KERNEL MODE BEFORE LEAVING
2636
2637                                  ;;*****************************************************************
2638                                  ;*TEST 30       TEST "W-BIT" SPECIAL CASES
2639                                  ;*
2640                                  ;*      THIS TEST CHECKS TWO SPECIAL CASES OF THE W-BIT. FIRST CASE IS
2641                                  ;*      THAT THE W-BIT SHOULD NOT SET IN PDR 7 WHEN WRITING TO
2642                                  ;*      STATUS REG SR0 (KERNEL PDR 7 IS USED).  SECOND CASE IS THAT
2643                                  ;*      THE W-BIT IS STILL SET IF THE "DATO" IS ABORTED DUE TO A
2644                                  ;*      TIMEOUT ERROR (KERNEL PDR6 & VIRTUAL ADDR 140000 ARE USED).
2645                                  ;*
2646                                  ;;*****************************************************************
2647  025626  000004                  TST30:  SCOPE
2648
2649  025630  004737  035100          1$:     JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2650  025634  012701  077406                  MOV     #77406,R1       ;PUT "W-BIT OFF" VALUE FOR PDR IN R1
2651  025640  012737  025646  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2652  025646  010137  172316          2$:     MOV     R1,KIPDR7       ;LOAD KERNEL PDR 7 TO CLEAR W-BIT
2653  025652  013700  177572                  MOV     SR0,R0          ;READ PRESENT CONTENTS OF STATUS REG. 0
2654  025656  010037  177572                  MOV     R0,SR0          ;WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
2655  025662  013702  172316                  MOV     KIPDR7,R2       ;READ CONTENTS OF KIPDR7 INTO R2
2656  025666  020102                          CMP     R1,R2           ;WAS W-BIT LEFT CLEARED?
2657  025670  001401                          BEQ     3$              ;BRANCH IF YES
2658  025672  104024                          ERROR   24              ;W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
2659                                                                  ;FOR TIGHTER SCOPE LOOP
2660                                                                  ;REPLACE ERROR CALL WITH
2661                                                                  ;"BR 2$" = 000765
2662  025674  012737  025674  001110  3$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2663  025702  010137  172314                  MOV     R1,KIPDR6       ;LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
2664  025706  012737  025720  000004          MOV     #4$,ERRVEC      ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2665  025714  005037  140000                  CLR     @#140000        ;CAUSE TIMEOUT ABORT THRU LOC. 4
2666  025720  012706  001100          4$:     MOV     #KERSTK,KSP     ;RESTORE THE STACK POINTER
2667  025724  013702  172314                  MOV     KIPDR6,R2       ;READ KIPDR6 INTO R2
2668  025730  052701  000100                  BIS     #100,R1         ;R1-77506
2669  025734  020102                          CMP     R1,R2           ;WAS W-BIT SET?
2670  025736  001401                          BEQ     5$              ;BRANCH IF YES
2671  025740  104025                          ERROR   25              ;W-BIT WAS NOT SET DURING A TIMEOUT ABORT
2672                                                                  ;FOR TIGHTER SCOPE LOOP
2673                                                                  ;REPLACE ERROR CALL WITH
2674                                                                  ;"BR 3$" = 000757
2675  025742  010137  172314          5$:     MOV     R1,KIPDR6       ;RESTORE KIPDR6 TO 77406
2676  025746  012737  001400  172354          MOV     #1400,KIPAR6    ;RESTORE KIPAR6 TO 1400
```

```
2677  025754  012737  002076  000004          MOV     #TIMERR,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2678  025762  012737  025630  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2679  025770  004737  035134                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
2680
2681                                  ;;*****************************************************************
2682                                  ;*
2683                                  ;*      THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2684                                  ;*      TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2685                                  ;*      ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2686                                  ;*      VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2687                                  ;*      THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2688                                  ;*      ARE CHECKED IN SR0.  SO THE SR0 AND SR2 LOGIC AND THE
2689                                  ;*      KT ERROR LOGIC ARE CHECKED.
2690                                  ;*
2691                                  ;;*****************************************************************
2692
2693                                  ;;*****************************************************************
2694                                  ;*TEST 31       NON-RESIDENT ABORT TEST (ACF=0&4)
2695                                  ;*
2696                                  ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2697                                  ;*      LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2698                                  ;*      USER MODES.  PDR 4 IS LOADED WITH ACF'S = 0&4 AND
2699                                  ;*      THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
2700                                  ;*
2701                                  ;;*****************************************************************
2702  025774  000004                  TST31:  SCOPE
2703
2704  025776  012700  000600          1$:     MOV     #600,R0         ;LOAD DATA FOR PAR'S INTO R0
2705  026002  010037  172346                  MOV     R0,KIPAR3       ;MAP KERNEL PAR'S 3&4 TO 12-16K
2706  026006  010037  172350                  MOV     R0,KIPAR4
2707  026012  010037  177646                  MOV     R0,UIPAR3       ;MAP USER PAR'S 3&4 TO 12-16K
2708  026016  010037  177650                  MOV     R0,UIPAR4
2709  026022  012737  077406  172306          MOV     #77406,KIPDR3   ;MAP KERNEL PDR 3  128 BLKS, READ-WRITE
2710  026030  012737  077406  177606          MOV     #77406,UIPDR3   ;MAP USER PDR 3  128 BLKS, READ-WRITE
2711  026036  012700  060000                  MOV     #60000,R0       ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2712  026042  012701  100000                  MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2713  026046  012703  100011                  MOV     #100011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., KERNEL, PG.4
2714  026052  012702  077400                  MOV     #77400,R2       ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2715  026056  012737  026120  000250  2$:     MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2716  026064  010237  172310                  MOV     R2,KIPDR4       ;LOAD ACF TEST VALUE INTO KIPDR4
2717  026070  010237  177610                  MOV     R2,UIPDR4       ;LOAD ACF TEST VALUE INTO UIPDR4
2718  026074  012737  026102  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2719  026102  005010                  3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
2720  026104  013737  177776  001176          MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
2721  026112  005211                  4$:     INC     (R1)            ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2722  026114  104026                          ERROR   26              ;MEM. MGMT. ABORT DID NOT OCCUR
2723                                                                  ;FOR TIGHTER SCOPE LOOP
2724                                                                  ;REPLACE ERROR CALL WITH
2725                                                                  ;"BR 3$" = 000772
2726  026116  000425                          BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2727  026120  062706  000004          5$:     ADD     #4,SP           ;RESTORE STACK POINTER
2728  026124  005710                          TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2729  026126  001401                          BEQ     6$              ;BRANCH IF YES
2730  026130  104027                          ERROR   27              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2731                                                                  ;FOR TIGHTER SCOPE LOOP
2732                                                                  ;REPLACE ERROR CALL WITH
```

```
2733                                                    ;"BR 3$" = 000764
2734  026132  013737  177572  001272  6$:   MOV   SR0,WASSR0     ;READ STATUS REGISTER 0
2735  026140  013737  177576  001274        MOV   SR2,WASSR2     ;READ STATUS REGISTER 2
2736  026146  020337  001272              CMP   R3,WASSR0      ;DID SR0 REPORT NON-RESIDENT ERROR CORRECTLY?
2737  026152  001401                      BEQ   7$             ;BRANCH IF YES
2738  026154  104030                      ERROR 30             ;SR0 DID NOT REPORT NON-RES. ERROR CORRECTLY
2739                                                    ;FOR TIGHTER SCOPE LOOP
2740                                                    ;REPLACE ERROR CALL WITH
2741                                                    ;"BR 3$" = 000752
2742  026156  012704  026112        7$:   MOV   #4$,R4         ;LOAD R4 WITH WHAT SR2 SHOULD READ
2743  026162  020437  001274              CMP   R4,WASSR2      ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2744  026166  001401                      BEQ   8$             ;BRANCH IF YES
2745  026170  104031                      ERROR 31             ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2746                                                    ;FOR TIGHTER SCOPE LOOP
2747                                                    ;REPLACE ERROR CALL WITH
2748                                                    ;"BR 3$" = 000744
2749  026172  042737  160000  177572  8$:   BIC   #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
2750  026200  032737  140000  001176        BIT   #140000,$TMP0  ;HAS ACF=084 BEEN TESTED IN USER YET
2751  026206  001006                      BNE   9$             ;BRANCH IF YES
2752  026210  012703  100151              MOV   #100151,R3     ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., USER, PG.4
2753  026214  012737  140000  177776      MOV   #140000,PSW    ;GO TO USER MODE
2754  026222  000715                      BR    2$             ;REPEAT TEST IN USER MODE
2755  026224  022702  077404        9$:   CMP   #77404,R2      ;HAS ACF=4 BEEN TESTED YET?
2756  026230  001407                      BEQ   10$            ;BRANCH IF YES
2757  026232  012702  077404              MOV   #77404,R2      ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2758  026236  012703  100011              MOV   #100011,R3     ;LOAD R3 WITH WHAT SR0 SHOULD READ-N.R.,KERNEL,PG. 4
2759  026242  005037  177776              CLR   PSW            ;GO BACK TO KERNEL MODE
2760  026246  000703                      BR    2$             ;GO BACK & TEST ACF=4 IN SAME MODE
2761  026250  005037  177776        10$:  CLR   PSW            ;GO BACK TO KERNEL MODE BEFORE LEAVING
2762  026254  012737  025776  001110      MOV   #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
2763  026262  012737  002150  000250      MOV   #MGMERR,MMVEC  ;RESTORE ADDRESS OF NORMAL MEMORY
2764                                                    ;MANAGEMENT ERROR ROUTINE TO MMVEC
2765
2766                                      ;;*****************************************************************
2767                                      ;*TEST 32       READ-ONLY ABORT TEST (ACF=2)
2768                                      ;*
2769                                      ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2770                                      ;*      LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2771                                      ;*      USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
2772                                      ;*      PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
2773                                      ;*
2774                                      ;;*****************************************************************
2775  026270  000004                TST32: SCOPE
2776  026272                        1$:                          ;KERNEL & USER PAR'S 3 & 4 AND PDR 3
2777                                                    ;ARE SETUP FROM LAST TEST
2778  026272  012700  060000              MOV   #60000,R0      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2779  026276  012701  100000              MOV   #100000,R1     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2780  026302  012703  020011              MOV   #20011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG.4
2781  026306  012702  077402              MOV   #77402,R2      ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
2782  026312  012737  026354  000250  2$:   MOV   #5$,MMVEC     ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2783  026320  010237  172310              MOV   R2,KIPDR4      ;LOAD ACF=2 INTO KIPDR4
2784  026324  010237  177610              MOV   R2,UIPDR4      ;LOAD ACF=2 INTO UIPDR4
2785  026330  012737  026336  001110      MOV   #3$,$LPERR     ;SET LOOP ON ERROR POINTER TO 3$
2786  026336  005010                3$:   CLR   (R0)           ;CLEAR PHYS. LOC. 60000 USING PDR3
2787  026340  013737  177776  001176      MOV   PSW,$TMP0      ;SAVE PSW IN CASE OF ERROR
2788  026346  005211                4$:   INC   (R1)           ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
```

```
2789  026350  104026                        ERROR   26              ;MEM. MGMT. ABORT DID NOT OCCUR
2790                                                                  ;FOR TIGHTER SCOPE LOOP
2791                                                                  ;REPLACE ERROR CALL WITH
2792                                                                  ;"BR 3$" = 000772
2793  026352  000425                        BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2794  026354  062706  000004          5$:   ADD     #4,SP           ;RESTORE STACK POINTER
2795  026360  005710                        TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2796  026362  001401                        BEQ     6$              ;BRANCH IF YES
2797  026364  104027                        ERROR   27              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2798                                                                  ;FOR TIGHTER SCOPE LOOP
2799                                                                  ;REPLACE ERROR CALL WITH
2800                                                                  ;"BR 3$" = 000764
2801  026366  013737  177572  001272  6$:   MOV     SR0,WASSR0      ;READ STATUS REG. 0
2802  026374  013737  177576  001274        MOV     SR2,WASSR2      ;READ STATUS REG. 2
2803  026402  020337  001272              CMP     R3,WASSR0       ;DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
2804  026406  001401                        BEQ     7$              ;BRANCH IF YES
2805  026410  104030                        ERROR   30              ;SR0 DID NOT REPORT R/O ERROR CORRECTLY
2806                                                                  ;FOR TIGHTER SCOPE LOOP
2807                                                                  ;REPLACE ERROR CALL WITH
2808                                                                  ;"BR 3$" = 000752
2809  026412  012704  026346          7$:   MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
2810  026416  020437  001274              CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2811  026422  001401                        BEQ     8$              ;BRANCH IF YES
2812  026424  104031                        ERROR   31              ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
2813                                                                  ;FOR TIGHTER SCOPE LOOP
2814                                                                  ;REPLACE ERROR CALL WITH
2815                                                                  ;"BR 3$" = 000744
2816  026426  042737  160000  177572  8$:   BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
2817  026434  032737  140000  001176        BIT     #140000,$TMP0   ;HAS ACF=2 BEEN TESTED IN USER MODE?
2818  026442  001006                        BNE     9$              ;BRANCH IF YES
2819  026444  012703  020151              MOV     #20151,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4
2820  026450  012737  140000  177776        MOV     #140000,PSW     ;GO TO USER MODE
2821  026456  000715                        BR      2$              ;REPEAT TEST IN USER MODE
2822  026460  005037  177776          9$:   CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
2823  026464  012737  026272  001110        MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2824  026472  012737  002150  000250        MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
2825                                                                  ;MANAGEMENT ERROR ROUTINE TO MMVEC.
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837                                  ;;*****************************************************************
2838                                  ;*
2839                                  ;*      THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
2840                                  ;*      COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
2841                                  ;*      AND STATUS LOGIC.  THE PAGE LENGTH FIELD (PLF) IN KERNEL
2842                                  ;*      PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
2843                                  ;*      ADDRESSES ARE READ.  WHILE USING BOTH UPWARD & DOWNWARD PAGE
2844                                  ;*      EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
```

```
                                        ;*      "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T.
                                        ;*
                                        ;*      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 INSTRUCTI"N THAT CAUSED THE ABORT
                                        ;*      IS LOCKED UP.
                                        ;*
                                        ;;*****************************************************************



                                        ;;*****************************************************************
                                        ;*TEST 33       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).
                                        ;*
                                        ;;*****************************************************************
026500  000004                  TST33:  SCOPE
026502  012737  077406  172306  1$:     MOV     #77406,KIPDR3   ;MAKE SURE PDR3 IS DESCRIBED AS R/W
026510  012737  077406  172312          MOV     #77406,KIPDR5   ;MAKE SURE PDR5 IS DESCRIBED AS R/W
026516  012700  026776                  MOV     #DALTB1,R0      ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
026522  012704  027014                  MOV     #PDRTB1,R4      ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
026526  012701  000006                  MOV     #6,R1           ;SET UP LOOP COUNTER.
026532  012737  026710  000250          MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
026540  012737  026552  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
026546  012706  001100                  MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP

                              ;TEST NON-ABORT CASES (VBA < OR = PLF)
026552  012437  172310        2$:     MOV     (R4)+,KIPDR4    ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
026556  005730                        TST     @(R0)+          ;ACCESS VIRTUAL ADDR. (VBA < OR = PLF)
                                                              ;NO ABORT SHOULD OCCUR!!!
026560  077104                        SOB     R1,2$           ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.

                              ;TEST ABORT CASES (VBA > PLF)
026562  012701  000005        3$:     MOV     #5,R1           ;SET UP LOOP COUNTER.
026566  012700  027032                MOV     #DALTB2,R0      ;DAL TABLE
026572  012704  027046                MOV     #PDRTB2,R4      ;PDR TABLE
026576  012737  026612  001110        MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
026604  012737  026624  000250        MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT

026612  012437  172310        4$:     MOV     (R4)+,KIPDR4    ;LOAD KIPDR4 WITH PAGE LENGHT VALUE
026616  005730                5$:     TST     @(R0)+          ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
026620  104033                        ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
                                                              ;FOR TIGHTER SCOPE LOOP
                                                              ;REPLACE ERROR CALL WITH
                                                              ;"BR 5$" = 000776
026622  000424                        BR      8$              ;BRANCH AROUND ABORT CHECKS
026624  012706  001100        6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
026630  013737  177572  001272        MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
026636  013737  177576  001274        MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
```

```
2901  026644  012702  040011            MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
2902  026650  020237  001272            CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
2903  026654  001401                    BEQ     7$              ;BRANCH IF YES
2904  026656  104034                    ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2905                                                            ;FOR TIGHTER SCOPE LOOP
2906                                                            ;REPLACE ERROR CALL WITH
2907                                                            ;"BR 5$" = 000757
2908  026660  012703  026616      7$:   MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
2909  026664  020337  001274            CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2910  026670  001401                    BEQ     8$              ;BRANCH IF YES
2911  026672  104035                    ERROR   35              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2912                                                            ;FOR TIGHTER SCOPE LOOP
2913                                                            ;REPLACE ERROR CALL WITH
2914                                                            ;"BR 5$" = 000751
2915  026674  042737  160000  177572  8$:   BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2916  026702  077135                    SOB     R1,4$           ;DONE?..NO - GET NEXT DAL & PDR PAIR
2917  026704  000137  026756            JMP     10$             ;YES...
2918  026710  012637  001266      9$:   MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
2919  026714  012637  001270            MOV     (KSP)+,TRAPPS
2920  026720  013737  177572  001272    MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
2921  026726  013737  177576  001274    MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
2922  026734  042737  160000  177572    BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2923  026742  104032                    ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2924                                                            ;FOR TIGHTER SCOPE LOOP
2925                                                            ;REPLACE ERROR CALL WITH
2926                                                            ;A "NOP" = 000240
2927  026744  013746  001270            MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
2928  026750  013746  001266            MOV     TRAPPC,-(KSP)
2929  026754  000002                    RTI                     ;RETURN FROM UNEXPECTED ABORT
2930
2931  026756  012737  026502  001110  10$:   MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2932  026764  012737  002150  000250        MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
2933                                                            ;ADDRESS TO M.M. TRAP VECTOR
2934  026772  000137  027062            JMP     TST34
2935
2936                                  ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
2937  026776  100000                  DALTB1: 100000
2938  027000  106100                          106100
2939  027002  102300                          102300
2940  027004  102500                          102500
2941  027006  113700                          113700
2942  027010  104600                          104600
2943  027012  117700                          117700
2944
2945                                  ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
2946  027014  000006                  PDRTB1: 000006
2947  027016  052006                          052006
2948  027020  045006                          045006
2949  027022  052006                          052006
2950  027024  074406                          074406
2951  027026  025006                          025006
2952  027030  077406                          077406
2953
2954                                  ;DAL TABLE (ABORT CASES)
2955  027032  100100                  DALTB2: 100100
2956  027034  110100                          110100
```

```
                                        116600
                                        112700
                                        117000
                                        117700

                                ;PDR TABLE (ABORT CASES)
                                PDRTB2: 000006
                                        030406
                                        046406
                                        042006
                                        073406
                                        077006

                                ;;*****************************************************************
                                ;*TEST 34       PAGE LENGTH FAULTS-DOWNWARD EXPANSION
                                ;*
                                ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
                                ;*      FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
                                ;*      ARE ACCESSED.  WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
                                ;*      NO PAGE ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS LESS THAN PDR <14:8>
                                ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED B' SR0 & SR2.
                                ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
                                ;*      (BIT 3) OF PDR4=1).
                                ;*
                                ;;*****************************************************************
027062  000004                  TST34:  SCOPE
027064  012700  027344          1$:     MOV     #DALTB3,R0      ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
027070  012704  027362          1$:     MOV     #PDRTB3,R4      ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
027074  012701  000006          1$:     MOV     #6,R1           ;SET UP LOOP COUNTER.
027100  012737  027256  000250  1$:     MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
027106  012737  027120  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
027114  012706  001100          1$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP

                                ;TEST NON-ABORT CASES (VBA > OR = PLF)
027120  012437  172310          2$:     MOV     (R4)+,KIPDR4    ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
027124  005730                  2$:     TST     @(R0)+          ;ACCESS VIRTUAL ADDR. (VBA > OR = PLF)
                                                                ;NO ABORT SHOULD OCCUR!!!
027126  077104                  2$:     SOB     R1,2$           ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.

                                ;TEST ABORT CASES (VBA < PLF)
027130  012701  000005          3$:     MOV     #5,R1           ;SET UP LOOP COUNTER.
027134  012700  027400          3$:     MOV     #DALTB4,R0      ;DAL TABLE
027140  012704  027414          3$:     MOV     #PDRTB4,R4      ;PDR TABLE
027144  012737  027160  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
027152  012737  027172  000250  3$:     MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT

027160  012437  172310          4$:     MOV     (R4)+,KIPDR4    ;LOAD KIPDR4 WITH PAGE LENGHT VALUE
027164  005730                  5$:     TST     @(R0)+          ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
027166  104033                          ERROR   33              ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 5$" = 000776
027170  000424                          BR      8$              ;BRANCH AROUND ABORT CHECKS
027172  012706  001100          6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
027176  013737  177572  001272          MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
```

```
3013  027204  013737  177576  001274          MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
3014  027212  012702  040011                  MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
3015  027216  020237  001272                  CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3016  027222  001401                          BEQ     7$              ;BRANCH IF YES
3017  027224  104034                          ERROR   34              ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3018                                                                  ;FOR TIGHTER SCOPE LOOP
3019                                                                  ;REPLACE ERROR CALL WITH
3020                                                                  ;"BR 5$" = 000757
3021  027226  012703  027164          7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
3022  027232  020337  001274                  CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3023  027236  001401                          BEQ     8$              ;BRANCH IF YES
3024  027240  104035                          ERROR   35              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3025                                                                  ;FOR TIGHTER SCOPE LOOP
3026                                                                  ;REPLACE ERROR CALL WITH
3027                                                                  ;"BR 5$" = 000751
3028  027242  042737  160000  177572  8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3029  027250  077135                          SOB     R1,4$           ;DONE?..NO - GET NEXT DAL & PDR PAIR
3030  027252  000137  027324                  JMP     10$             ;YES...
3031  027256  012637  001266          9$:     MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3032  027262  012637  001270                  MOV     (KSP)+,TRAPPS
3033  027266  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
3034  027274  013737  177576  001274          MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3035  027302  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3036  027310  104032                          ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3037                                                                  ;FOR TIGHTER SCOPE LOOP
3038                                                                  ;REPLACE ERROR CALL WITH
3039                                                                  ;A "NOP" = 000240
3040  027312  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3041  027316  013746  001266                  MOV     TRAPPC,-(KSP)
3042  027322  000002                          RTI                     ;RETURN FROM UNEXPECTED ABORT
3043
3044  027324  012737  027064  001110  10$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3045  027332  012737  002150  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3046                                                                  ;ADDRESS TO M.M. TRAP VECTOR
3047  027340  000137  027430                  JMP     TST35
3048
3049                                  ;DAL TABLE FOR DOWNWARD EXPANSION (NON-ABORT CASES)
3050  027344  117700                  DALTB3: 117700
3051  027346  111600                          111600
3052  027350  115400                          115400
3053  027352  115200                          115200
3054  027354  104000                          104000
3055  027356  113100                          113100
3056  027360  100000                          100000
3057
3058                                  ;PDR TABLE (NON-ABORT CASES)
3059  027362  077416                  PDRTB3: 77416
3060  027364  025416                          25416
3061  027366  032416                          32416
3062  027370  025416                          25416
3063  027372  003016                          03016
3064  027374  052416                          52416
3065  027376  000016                          00016
3066
3067                                  ;DAL TABLE (ABORT CASES)
3068  027400  117600                  DALTB4: 117600
```

```
3069  027402  107600                          107600
3070  027404  101100                          101100
3071  027406  105000                          105000
3072  027410  100700                          100700
3073  027412  100000                          100000
3074
3075                                  ;PDR TABLE (ABORT CASES)
3076  027414  077416                  PDRTB4: 77416
3077  027416  047016                          47016
3078  027420  031016                          31016
3079  027422  035416                          35416
3080  027424  004016                          04016
3081  027426  000416                          00416
3082
3083
3084
3085                                  ;;*****************************************************************
3086                                  ;*TEST 35       SR2 BIT TEST
3087                                  ;*
3088                                  ;*      THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
3089                                  ;*      CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 100000
3090                                  ;*      TO 110000 (PHYSICAL ADDRESSES 060000-070000).  KIPDR4 IS USED TO EXECUTE
3091                                  ;*      THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
3092                                  ;*      010727  MOV     PC,(PC)+  ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
3093                                  ;*      000000                    ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
3094                                  ;*      000137  JMP     @#3$      ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
3095                                  ;*      (ADDR. OF 3$)             ;IN CASE A R/O ABORT DOES NOT OCCUR,
3096                                  ;*                                ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
3097                                  ;*
3098                                  ;;*****************************************************************
3099  027430  000004                  TST35:  SCOPE
3100  027432  012737  000600  172346  1$:     MOV     #600,KIPAR3       ;BE SURE PAR3 IS MAPPED TO 12-16K
3101  027440  012737  000600  172350          MOV     #600,KIPAR4       ;BE SURE PAR4 IS MAPPED TO 12-16K
3102  027446  012737  077406  172306          MOV     #77406,KIPDR3     ;MAP PAGE 3  128 BLOCKS, R/W
3103  027454  012737  077402  172310          MOV     #77402,KIPDR4     ;MAP PAGE 4  128 BLOCKS, READ-ONLY
3104  027462  012700  060002                  MOV     #60002,R0         ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
3105  027466  012701  100002                  MOV     #100002,R1        ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
3106  027472  012737  027526  000250          MOV     #3$,MMVEC         ;SET M.M. TRAP VECTOR TO 3$
3107  027500  012737  027506  001110          MOV     #2$,$LPERR        ;SET LOOP ON ERROR POINTER TO 2$
3108  027506  012720  010727          2$:     MOV     #010727,(R0)+     ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR.
3109  027512  005020                          CLR     (R0)+             ;       REACHED THRU PDR/PAR 4.
3110  027514  012720  000137                  MOV     #000137,(R0)+     ;LOAD "JMP @#3$" INSTRUCTION AT VIRT. ADDR.
3111  027520  012710  027526                  MOV     #3$,(R0)          ;       IN CASE R/O VIOL. DOES NOT ABORT
3112  027524  010107                          MOV     R1,PC             ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3113  027526  012706  001100          3$:     MOV     #KERSTK,KSP       ;RESTORE STACK POINTER
3114  027532  013737  177576  001274          MOV     SR2,WASSR2        ;READ CONTENTS OF STATUS REG 2
3115  027540  020137  001274                  CMP     R1,WASSR2         ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
3116  027544  001401                          BEQ     4$                ;BRANCH IF YES
3117  027546  104036                          ERROR   36                ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
3118                                                                    ;FOR TIGHTER SCOPE LOOP
3119                                                                    ;REPLACE ERROR CALL WITH
3120                                                                    ;"BR 2$" = 000757
3121  027550  042737  160000  177572  4$:     BIC     #160000,SR0       ;CLEAR THE ERROR BITS IN SR0
3122  027556  060101                          ADD     R1,R1             ;SETUP TO FORM NEXT VIRTUAL ADDRESS
3123  027560  010100                          MOV     R1,R0             ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD
3124  027562  052701  100000                  BIS     #100000,R1        ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
```

```
3125  027566  052700  060000                    BIS     #60000,R0       ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD
3126  027572  020127  110000                    CMP     R1,#110000      ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?
3127  027576  101743                            BLOS    2$              ;BRANCH IF NO
3128
3129  027600  012737  027432  001110  5$:       MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3130  027606  012737  077406  172310            MOV     #77406,KIPDR4   ;RESTORE PDR4 TO R/W ACCESS
3131  027614  012737  002150  000250            MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL M.M.
3132                                                                    ;TRAP HANDLER TO M.M. VECTOR
3133
3134
3135                                    ;;*****************************************************************
3136                                    ;*TEST 36       MORE CHECKS OF SR0 & SR2
3137                                    ;*
3138                                    ;*      THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SR0 & SR2 LOGIC.
3139                                    ;*      FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS
3140                                    ;*      PROGRAM COUNTER.  ALSO SR0 & SR2 ARE LOCKED UP BY A PAGE LENGTH
3141                                    ;*      ABORT, THEN WITHOUT CLEARING SR0'S ERROR BITS, A R/O ABORT IS CAUSED.
3142                                    ;*      SR0 & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
3143                                    ;*      INFORMATION ABOUT THE PAGE LENGTH ABORT SOULD STILL BE LOCKED UP.
3144                                    ;*      IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SR0 IS CLEARED
3145                                    ;*      AND SR2 IS UNLOCKED BY A RESET.  AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
3146                                    ;*      SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
3147                                    ;*
3148                                    ;*****************************************************************
3149  027622  000004                    TST36:  SCOPE
3150  027624  012737  000600  172352    1$:     MOV     #600,KIPAR5     ;MAP KERNEL PAGE 5 TO 12-16K
3151  027632  012737  000406  172310            MOV     #406,KIPDR4     ;SETUP PDR4 FOR PAGE LENGTH ABORT
3152  027640  012737  077402  172312            MOV     #77402,KIPDR5   ;SETUP PDR5 FOR R/O ABORT
3153  027646  012737  027654  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3154  027654  013737  177576  001274  2$:       MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
3155  027662  012701  027654                    MOV     #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
3156  027666  020137  001274                    CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
3157  027672  001401                            BEQ     3$              ;BRANCH IF YES
3158  027674  104041                            ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3159                                                                    ;FOR TIGHTER SCOPE LOOP
3160                                                                    ;REPLACE ERROR CALL WITH
3161                                                                    ;"BR 2$" = 000767
3162  027676  012737  027704  001110  3$:       MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3163  027704  013737  177576  001274  4$:       MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
3164  027712  012701  027704                    MOV     #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
3165  027716  020137  001274                    CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 4$
3166  027722  001401                            BEQ     5$              ;BRANCH IF YES
3167  027724  104041                            ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3168                                                                    ;FOR TIGHTER SCOPE LOOP
3169                                                                    ;REPLACE ERROR CALL WITH
3170                                                                    ;"BR 4$" = 000767
3171  027726  012737  027734  001110  5$:       MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
3172  027734  012737  027752  000250  6$:       MOV     #7$,MMVEC       ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
3173  027742  005037  001200                    CLR     $TMP1           ;CLEAR ERROR INDICATOR
3174  027746  005237  100500                    INC     @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
3175  027752  012706  001100            7$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
3176  027756  013737  177572  001176            MOV     SR0,$TMP0       ;SAVE SR0'S INFORMATION ON PG. LGTH. ABORT
3177  027764  013737  177576  001202            MOV     SR2,$TMP2       ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3178  027772  012737  030004  000250            MOV     #8$,MMVEC       ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
3179  030000  005237  120000                    INC     @#120000        ;CAUSE R/O ABORT - TRAP TO 8$
3180  030004  012706  001100            8$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
```

```
3181  030010  013737  177572  001272          MOV     SR0,WASSR0      ;READ SR0 FOLLWOING SECOND KT ABORT
3182  030016  013737  177576  001274          MOV     SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
3183  030024  023737  001176  001272          CMP     $TMP0,WASSR0    ;IS SR0 STILL HOLDING INFO ON FIRST ABORT?
3184  030032  001402                          BEQ     9$              ;BRANCH IF YES
3185  030034  005237  001200                  INC     $TMP1           ;SET ERROR INDICATOR
3186  030040  023737  001202  001274  9$:     CMP     $TMP2,WASSR2    ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
3187  030046  001402                          BEQ     10$             ;BRANCH IF YES
3188  030050  005237  001200                  INC     $TMP1           ;SET ERROR INDICATOR
3189  030054  005737  001200          10$:    TST     $TMP1           ;WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
3190  030060  001401                          BEQ     11$             ;BRANCH IF NO
3191  030062  104037                          ERROR   37              ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3192                                                                  ;FOR TIGHTER SCOPE LOOP
3193                                                                  ;REPLACE ERROR CALL WITH
3194                                                                  ;"BR 6$" = 000726
3195  030064  005037  001200          11$:    CLR     $TMP1           ;CLEAR ERROR INDICATOR
3196  030070  000005                          RESET                   ;EXECUTE A RESET, APPLYING AN "INIT"
3197  030072  013737  177572  001272          MOV     SR0,WASSR0      ;READ SR0
3198  030100  005737  001272                  TST     WASSR0          ;WAS SR0 CLEARED BY THE RESET?
3199  030104  001402                          BEQ     12$             ;BRANCH IF YES
3200  030106  005237  001200                  INC     $TMP1           ;SR0 NOT CLEARED BY A RESET
3201  030112  013737  177576  001274  12$:    MOV     SR2,WASSR2      ;READ SR2
3202  030120  022737  030112  001274          CMP     #12$,WASSR2     ;WAS SR2 UNLOCKED BY A RESET?
3203  030126  001402                          BEQ     13$             ;BRANCH IF YES
3204  030130  005237  001200                  INC     $TMP1           ;SR2 NOT UNLOCKED BY A RESET
3205  030134  005737  001200          13$:    TST     $TMP1           ;WERE SR0 & SR2 BOTH "RESET" BY A RESET?
3206  030140  001401                          BEQ     14$             ;BRANCH IF YES
3207  030142  104040                          ERROR   40              ;SR0 OR SR2 NOT "RESET" BY A RESET
3208                                                                  ;FOR TIGHTER SCOPE LOOP
3209                                                                  ;REPLACE ERROR CALL WITH
3210                                                                  ;"BR 6$" = 000676
3211  030144  012737  000001  177572  14$:    MOV     #1,SR0          ;TURN MEMORY MANAGEMENT BACK ON
3212  030152  013737  177576  001274  15$:    MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3213  030160  012701  030152                  MOV     #15$,R1         ;PUT EXPECTED VIRTUAL PC IN R1
3214  030164  020137  001274                  CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3215  030170  001401                          BEQ     16$             ;BRANCH IF YES
3216  030172  104041                          ERROR   41              ;SR2 NOT TRACKING CORRECTLY
3217                                                                  ;FOR TIGHTER SCOPE LOOP
3218                                                                  ;REPLACE ERROR CALL WITH
3219                                                                  ;"BR 6$" = 000663
3220  030174  012737  027624  001110  16$:    MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3221  030202  012737  077406  172310          MOV     #77406,KIPDR4   ;RESET PDR4 TO 128 BLKS, R/W
3222  030210  012737  077406  172312          MOV     #77406,KIPDR5   ;RESET PDR5 TO 128 BLKS, R/W
3223  030216  012737  002150  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
3224                                                                  ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
3225
3226
3227                                  ;;*****************************************************************
3228                                  ;*TEST 37       USER ABORT PICKS UP KERNEL SPACE VECTOR
3229                                  ;*
3230                                  ;*      THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
3231                                  ;*      USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
3232                                  ;*      KERNEL SPACE.  USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
3233                                  ;*      THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
3234                                  ;*      WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
3235                                  ;*      SHOULD BE PICKED UP FROM LOC. 000004.  A TIMEOUT ERROR IS USED
3236                                  ;*      TO CAUSE A TRAP TO "4".
```

```
                                        ;*
                                        ;;*****************************************************************
030224  000004                          TST37:  SCOPE
030226  004737  035100                  1$:     JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
030232  012737  030240  001110                  MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
030240  005037  177776                  2$:     CLR     PSW             ;GO TO KERNEL MODE
030244  012706  001100                          MOV     #KERSTK,KSP     ;SETUP KERNEL STACK PTR.
030250  012737  000600  177640                  MOV     #600,U1PAR0     ;MAP USER PAGE 0 TO 12K
030256  012737  030340  000004                  MOV     #4$,@#4         ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
030264  012737  000340  000006                  MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
030272  012737  140000  177776                  MOV     #140000,PSW     ;GO TO USER MODE
030300  012706  000700                          MOV     #USESTK,USP     ;SETUP USER STACK PTR.
030304  012737  030324  000004                  MOV     #3$,@#4         ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
030312  012737  000340  000006                  MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
030320  005737  160000                          TST     160000          ;CAUSE TIMEOUT ERROR TRAP TO "4"
                                                                        ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
                                                                        ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
030324  013701  177776                  3$:     MOV     PSW,R1          ;SAVE PSW FOR ERROR
030330  010602                                  MOV     SP,R2           ;SAVE VALUE OF STACK POINTER FOR ERROR
030332  005037  177776                          CLR     PSW             ;BE SURE BACK IN KERNEL MODE
030336  104042                                  ERROR   42              ;DID NOT TRAP THRU KERNEL SPACE
                                                                        ;FOR TIGHTER SCOPE LOOP
```

```
3259                                                    ;REPLACE ERROR CALL WITH
3260                                                    ;"BR 2$" = 000740
3261  030340  005037  177776          4$:     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
3262  030344  012706  001100                  MOV     #KERSTK,KSP     ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3263  030350  005037  177640                  CLR     UIPAR0          ;REMAP USER PAGE 0 TO 0-4K
3264  030354  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
3265  030362  012706  000700                  MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
3266  030366  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
3267  030372  012737  002076  000004          MOV     #TIMERR,@#4     ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3268  030400  012737  030226  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3269  030406  004737  035134                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
3270
3271                                          ;;*****************************************************************
3272                                          ;*TEST 40       RTI IN USER MODE DOES NOT CHANGE PSW
3273                                          ;*
3274                                          ;*      THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3275                                          ;*      MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3276                                          ;*
3277                                          ;;*****************************************************************
3278  030412  000004                  TST40:  SCOPE
3279
3280  030414  012737  030426  001110  1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3281  030422  012702  170000                  MOV     #170000,R2      ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
3282  030426  010237  177776          2$:     MOV     R2,PSW          ;GO TO USER MODE-PRIORITY 0
3283  030432  012746  000340                  MOV     #340,-(SP)      ;PUT A NEW PSW (PRIORITY=7) ON STACK
3284  030436  012746  030444                  MOV     #3$,-(SP)       ;PUT NEW PC ON THE STACK
3285  030442  000002                          RTI                     ;DO AN RTI FROM USER MODE
3286  030444  013701  177776          3$:     MOV     PSW,R1          ;READ NEW PSW INTO R1
3287  030450  042701  007437                  BIC     #7437,R1        ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3288  030454  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
3289  030460  020201                          CMP     R2,R1           ;DID PSW STAY IN USER, PRIORITY=0?
3290  030462  001401                          BEQ     4$              ;BRANCH IF YES
3291  030464  104060                          ERROR   60              ;PSW CHANGED BY AN RTI FROM USER
3292                                                                  ;FOR A TIGHTER SCOPE LOOP
3293                                                                  ;REPLACE ERROR CALL WITH
3294                                                                  ;"BR=2$" = 000760
3295  030466  012737  030414  001110  4$:     MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3296
3297
```

```
                                ;;*****************************************************************
                                ;*TEST 41       KT ERROR SERVICED BEFORE TIMEOUT ERROR
                                ;*
                                ;*      THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
                                ;*      WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES A TIMEOUT
                                ;*      ERROR FIRST, THE TIMEOUT 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
                                ;*
                                ;;*****************************************************************
030474  000004                  TST41:  SCOPE
030476  012705  077006          1$:     MOV     #77006,R5       ;LOAD PDR7 DATA INTO R5
030502  010537  172316                  MOV     R5,KIPDR7       ;MAP PAGE 7 R/W  PLF=176
030506  012737  030534  000004          MOV     #3$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 3$
030514  012737  030536  000250          MOV     #4$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$
030522  012737  030530  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
030530  005237  177700          2$:     INC     @#177700        ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT
030534  104043                  3$:     ERROR   43              ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000776
030536  012706  001100          4$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER TRAPPING
030542  013737  177572  001272          MOV     SR0,WASSR0      ;READ STATUS REG.0
030550  013737  177576  001274  5$:     MOV     SR2,WASSR2      ;READ STATUS REG. 2
030556  012700  040017                  MOV     #40017,R0       ;LOAD EXPECTED SR0 CONTENTS INTO R0
030562  020037  001272                  CMP     R0,WASSR0       ;SR0 PLF ERROR BIT SET?
030566  001401                          BEQ     6$              ;BRANCH IF YES
030570  104044                          ERROR   44              ;SR0 DIDN'T REPORT PLF ERROR
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000741
030572  012701  030530          6$:     MOV     #2$,R1          ;LOAD EXPECTED SR2 CONTENTS INTO R1
030576  020137  001274                  CMP     R1,WASSR2       ;WAS SR2 LOCKED BY PLF ABORT?
030602  001401                          BEQ     7$              ;BRANCH IF YES
030604  104044                          ERROR   44              ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 2$" = 000741
030606  042737  160000  177572  7$:     BIC     #160000,SR0     ;CLEAR ERROR BITS THAT WERE SET IN SR0
030614  012737  002076  000004          MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
030622  012737  002150  000250          MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
030630  012737  077406  172316          MOV     #77406,KIPDR7   ;REMAP PAGE 7 TO READ/WRITE PLF=177
030636  012737  030476  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$

                                ;;*****************************************************************
                                ;*TEST 42       PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT 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 A TIMEOUT ERROR.  DURING A "DOUBLE ERROR"
                                ;*      SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
                                ;*      FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
                                ;*      NOT THE PSW PRESENT BEFORE THE FIRST TRAP.  SR0 AND SR2
```

```
                                        ;*      SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
                                        ;*
                                        ;*      NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
                                        ;*      WILL BE SET IN THE PSW THAT IS SAVED.
                                        ;*
                                        ;;*****************************************************************
030644  000004                  T5T42:  SCOPE
030646  004737  035100          1$:     JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
030652  012737  000600  177646          MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12-16K
030660  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12-16K
030666  012737  077402  177606          MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
030674  012737  077406  177610          MOV     #77406,UIPDR4   ;MAP USER PAGE 4 READ/WRITE
030702  012737  030756  000004          MOV     #4$,@#4         ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
030710  012737  140017  000006          MOV     #140017,@#6     ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
030716  012737  030756  000250          MOV     #4$,@#250       ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
030724  012737  000340  000252          MOV     #340,@#252      ;LOAD A KERNEL PSW IN MMVEC+2
030732  012737  030740  001110          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
030740  012737  140000  177776  2$:     MOV     #140000,PSW     ;GO TO USER MODE
030746  012706  100002                  MOV     #100002,USP     ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
030752  005737  177700          3$:     TST     @#177700        ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
                                                                ;R/O ERROR WHEN TRY TO SAVE OLD PC
030756  016601  000002          4$:     MOV     2(KSP),R1       ;PUT PSW SAVED ON KERNEL STACK INTO R1
030762  011603                          MOV     (KSP),R3        ;PUT PC SAVED ON KERNEL STACK INTO R3
030764  013737  177572  001272          MOV     SR0,WASSR0      ;READ THE CONTENTS OF M.M. STATUS REG. 0
030772  013737  177576  001274          MOV     SR2,WASSR2      ;READ THE CONTENTS OF M.M. STATUS REG. 2
031000  042737  160000  177572          BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
031006  005037  177776                  CLR     PSW             ;BE SURE IN KERNEL MODE
031012  012706  001100                  MOV     #KERSTK,KSP     ;RESTORE KERNEL STACK POINTER
031016  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
031024  012706  000700                  MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
031030  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
031034  005037  001176                  CLR     $TMP0           ;CLEAR ERROR INDICATOR
031040  020127  170017                  CMP     R1,#170017      ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
                                                                ;TIMEOUT TRAP FROM ERRVEC+2?
                                                                ;VALUE 170017 = PSW FROM LOC. 6 WITH
                                                                ;PREVIOUS MODE BITS = USER
031044  001402                          BEQ     5$              ;BRANCH IF YES
031046  005237  001176                  INC     $TMP0           ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
```

```
3392  031052  020327  030756          5$:     CMP     R3,#3$+4        ;WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
3393                                                                  ;SAVED ON THE STACK?
3394  031056  001402                          BEQ     6$              ;BRANCH IF YES
3395  031060  005237  001176                  INC     $TMP0           ;WRONG PC SAVED DURING TRAP SEQUENCE
3396  031064  023727  001272  020147  6$:     CMP     WASSR0,#20147   ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
3397  031072  001402                          BEQ     7$              ;BRANCH IF YES
3398  031074  005237  001176                  INC     $TMP0           ;SR0 DID NOT REPORT R/O ABORT
3399  031100  023727  001274  030752  7$:     CMP     WASSR2,#3$      ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3400                                                                  ;INSTRUCTION SUCCESSFULLY FETCHED?
3401  031106  001402                          BEQ     8$              ;BRANCH IF YES
3402  031110  005237  001176                  INC     $TMP0           ;SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
3403  031114  005737  001176          8$:     TST     $TMP0           ;ANY "ERRORS" DURING TRAP SEQUENCE?
3404  031120  001401                          BEQ     9$              ;BRANCH IF NO
3405  031122  104045                          ERROR   45              ;THE WRONG PC OR PSW WERE SAVED
3406                                                                  ;OR SR0 OR SR2 DID NOT REPORT R/O
3407                                                                  ;ERROR DURING TIMEOUT - KT TRAP
3408                                                                  ;SEQUENCE
3409                                                                  ;FOR TIGHTER SCOPE LOOP
3410                                                                  ;REPLACE ERROR CALL WITH
3411                                                                  ;"BR 2$" = 000710
3412  031124  012737  002076  000004  9$:     MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3413  031132  012737  000340  000006          MOV     #340,@#6        ;RELOAD ERRVEC+2 WITH KERNEL PSW
3414  031140  012737  002150  000250          MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3415  031146  012737  077406  177606          MOV     #77406,UIPDR3   ;REMAP USER PAGE 3 READ/WRITE
3416  031154  012737  030646  001110          MOV     #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3417  031162  004737  035134                  JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
3418                                  ;;*****************************************************************
3419                                  ;*
3420                                  ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3421                                  ;*      THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS.
3422                                  ;*
3423                                  ;;*****************************************************************
3424
3425
3426                                  ;;*****************************************************************
3427                                  ;*TEST 43       MOVE FROM PREVIOUS (USER) I-SPACE
3428                                  ;*
3429                                  ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3430                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3431                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3432                                  ;*
3433                                  ;*
3434                                  ;*      IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3435                                  ;*      WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
3436                                  ;*
3437                                  ;;*****************************************************************
3438  031166  000004                  TST43:  SCOPE
3439  031170  005037  172340          1$:     CLR     KIPAR0          ;MAP KERNEL PAGE 0 TO 0-4K
3440  031174  012737  000200  172342          MOV     #200,KIPAR1     ;MAP KERNEL PAGE 1 TO 4-8K
3441  031202  012737  000400  172344          MOV     #400,KIPAR2     ;MAP KERNEL PAGE 2 TO 8-12K
3442  031210  012737  000600  172346          MOV     #600,KIPAR3     ;MAP KERNEL PAGE 3 TO 12-16K
3443  031216  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
3444  031224  012737  007600  172356          MOV     #7600,KIPAR7    ;MAP KERNEL PAGE 7 TO THE I/O PAGE
3445  031232  012700  077406                  MOV     #77406,R0       ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3446                                                                  ;READ/WRITE, LENGTH 200 BLOCKS
3447  031236  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
```

```
3448  031242  012701  172300            MOV     #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3449  031246  010021            2$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3450  031250  077202                    SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL PDRS LOADED
3451  031252  012702  000010            MOV     #10,R2          ;SET LOOP COUNTER TO 8
3452  031256  012701  177600            MOV     #UIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
3453  031262  010021            3$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3454  031264  077202                    SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL PDRS LOADED
3455  031266  012737  000000  177640    MOV     #000,UIPAR0     ;MAP USER I PAGE 0 TO 0-4K
3456  031274  012737  000200  177642    MOV     #200,UIPAR1     ;MAP USER I PAGE 1 TO 4-8K
3457  031302  012737  000400  177644    MOV     #400,UIPAR2     ;MAP USER I PAGE 2 TO 8-12K
3458  031310  012737  000600  177646    MOV     #600,UIPAR3     ;MAP USER I PAGE 3 TO 12-16K
3459  031316  012737  007600  177656    MOV     #7600,UIPAR7    ;MAP USER I PAGE 7 TO THE I/O PAGE
3460  031324  012737  031332  001110    MOV     #4$,$LPERR      ;SET LOOP ON ERROR TO 4$
3461  031332                    4$:
3462  031332  012737  077406  172310    MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3463  031340  012737  000600  172350    MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3464  031346  012737  000600  177650    MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3465  031354  012700  036514            MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3466  031360  010037  100000            MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3467  031364  012737  031766  000250    MOV     #23$,MMVEC      ;SET M.M. VECTOR TO 23$
3468  031372  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3469                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3470                                    ;
3471  031376  012737  031404  001110    MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3472  031404  012737  030340  177776  5$:MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3473  031412  006506            6$:     MFPI    USP             ;PUT USER STACK POINTER ON KERNEL
3474                                                            ;STACK
3475  031414  022706  001100            CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
```

```
 3476  031420  001407                    BEQ     7$              ;BRANCH IF NOTHING WAS PUSHED
 3477  031422  012600                    MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
 3478  031424  012701  000700            MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
```

```
3479  031430  020001                        CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3480  031432  001403                        BEQ     8$              ;BRANCH IF YOU DID
3481  031434  104046                        ERROR   46              ;WRONG THING WAS PUSHED ON STACK
3482                                                                ;FOR TIGHTER SCOPE LOOP
3483                                                                ;REPLACE ERROR CALL WITH
3484                                                                ;"BR 5$" = 000763
3485  031436  000401                        BR      8$              ;BRANCH TO NEXT TRY
3486  031440  104050                7$:     ERROR   50              ;NOTHING PUSHED ON STACK
3487                                                                ;FOR TIGHTER SCOPE LOOP
3488                                                                ;REPLACE ERROR CALL WITH
3489                                                                ;"BR 5$" = 000761
3490  031442                        8$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3491  031442  012737  031454  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3492  031450  012700  036514                MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3493  031454  012737  030340  177776  9$:   MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3494  031462  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3495  031466  006512                        MFPI    (R2)            ;READ FROM PHYSICAL 60000
3496  031470  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3497  031472  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3498  031474  001401                        BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3499  031476  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3500                                                                ;FOR TIGHTER SCOPE LOOP
3501                                                                ;REPLACE ERROR CALL WITH
3502                                                                ;"BR 9$" = 000766
3503  031500                        10$:    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3504  031500  012737  031506  001110        MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3505  031506  012737  030340  177776  11$:  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3506  031514  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3507  031520  006522                        MFPI    (R2)+           ;READ FROM PHYSICAL 60000
3508  031522  012601                        MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3509  031524  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3510  031526  001401                        BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3511  031530  104046                        ERROR   46              ;WRONG DATA WAS FETCHED
3512                                                                ;FOR TIGHTER SCOPE LOOP
3513                                                                ;REPLACE ERROR CALL WITH
3514                                                                ;"BR 11$" = 000766
3515  031532                        12$:    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
```

```
3516  031532  012737  031540  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3517  031540  012737  030340  177776  13$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3518  031546  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
3519  031552  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3520  031554  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3521  031556  001401                          BEQ     14$             ;BRANCH IF CORRECT DATA WAS FETCHED
3522  031560  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3523                                                                  ;FOR TIGHTER SCOPE LOOP
3524                                                                  ;REPLACE ERROR CALL WITH
```

```
3525                                                    ;"BR 13$" = 000767
3526  031562                            14$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3527  031562  012737  031570  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
3528  031570  012737  030340  177776  15$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3529  031576  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3530  031602  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 60000
3531  031604  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3532  031606  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3533  031610  001401                          BEQ     16$             ;BRANCH IF CORRECT DATA WAS FETCHED
3534  031612  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3535                                                                  ;FOR TIGHTER SCOPE LOOP
3536                                                                  ;REPLACE ERROR CALL WITH
3537                                                                  ;"BR 15$" = 000766
3538  031614                            16$:
3539                                          ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3540                                          ;
3541  031614  012737  031622  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
3542  031622  012737  030340  177776  17$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3543  031630  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3544  031636  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
3545  031642  006552                          MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
3546  031644  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3547  031646  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3548  031650  001401                          BEQ     18$             ;BRANCH IF CORRECT DATA WAS FETCHED
3549  031652  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3550                                                                  ;FOR TIGHTER SCOPE LOOP
3551                                                                  ;REPLACE ERROR CALL WITH
3552                                                                  ;"BR 17$" = 000763
3553  031654                            18$:    ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3554                                          ;
3555  031654  012737  031662  001110          MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
3556  031662  012737  030340  177776  19$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3557  031670  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3558  031672  006562  100000                  MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
3559  031676  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3560  031700  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3561  031702  001401                          BEQ     20$             ;BRANCH IF CORRECT DATA WAS FETCHED
3562  031704  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3563                                                                  ;FOR TIGHTER SCOPE LOOP
3564                                                                  ;REPLACE ERROR CALL WITH
3565                                                                  ;"BR 19$" = 000766
3566  031706                            20$:    ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3567                                          ;
3568  031706  012737  031714  001110          MOV     #21$,$LPERR     ;SET LOOP ON ERROR POINTER TO 21$
3569  031714  012737  030340  177776  21$:    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3570  031722  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
3571  031730  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3572  031734  006572  000000                  MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
3573                                                                  ;ADDRESS OF 60000
3574  031740  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3575  031742  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3576  031744  001401                          BEQ     22$             ;BRANCH IF CORRECT DATA WAS FETCHED
3577  031746  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3578                                                                  ;FOR TIGHTER SCOPE LOOP
3579                                                                  ;REPLACE ERROR CALL WITH
3580                                                                  ;"BR 21$" = 000762
```

```
3581  031750  012737  002150  000250  22$:    MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3582  031756  012737  031170  001110          MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3583  031764  000423                          BR      TST44           ;;BRANCH TO NEXT TEST
3584
3585
3586  031766  012637  001266          23$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3587  031772  012637  001270                  MOV     (KSP)+,TRAPPS
3588  031776  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3589  032004  013737  177576  001274          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3590  032012  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
3591  032020  104051                          ERROR   51              ;TRIED TO READ NON-RESIDENT PAGE
3592                                                                  ;FOR TIGHTER SCOPE LOOP
3593                                                                  ;REPLACE ERROR CALL WITH
3594                                                                  ;A "NOP" = 000240
3595  032022  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3596  032026  013746  001266                  MOV     TRAPPC,-(KSP)
3597  032032  000002                          RTI
3598
3599
3600                                  ;;*****************************************************************
3601                                  ;*TEST 44       MOVE TO PREVIOUS (USER) I-SPACE
3602                                  ;*
3603                                  ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3604                                  ;*      PREVIOUS MODE IS CLOCKED CORRECTLY
3605                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3606                                  ;*
3607                                  ;*
3608                                  ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3609                                  ;*      WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3610                                  ;*
3611                                  ;;*****************************************************************
3612  032034  000004                  TST44:  SCOPE
3613  032036  012737  077406  172310  1$:     MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3614  032044  012737  077406  177610          MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
3615  032052  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3616  032060  012737  000600  177650          MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3617  032066  012737  032646  000250          MOV     #20$,MMVEC      ;SET M.M. VECTOR TO 20$
3618                                          ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3619                                          ;
3620  032074  012737  030340  177776  2$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3621  032102  012746  007777                  MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3622  032106  006606                          MTPI    USP             ;LOAD USER STACK POINTER
3623  032110  006506                          MFPI    USP             ;READ USER STACK POINTER
3624  032112  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3625  032114  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
3626  032120  001401                          BEQ     3$              ;BRANCH IF IT WAS
3627  032122  104050                          ERROR   50              ;USER STACK POINTER NOT CHANGED
3628                                                                  ;FOR TIGHTER SCOPE LOOP
3629                                                                  ;REPLACE ERROR CALL WITH
3630                                                                  ;"BR 2$" = 000764
3631  032124  012737  030340  177776  3$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3632  032132  012746  000700                  MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER S. POINT
3633  032136  006606                          MTPI    USP             ;RESTORE USER STACK POINTER
3634  032140                          4$:     ;THIS WILL TEST DSTM = 1 MTPI.
3635  032140  012737  032156  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3636  032146  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
```

```
3637  032152  012700  125252            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3638  032156  010046            5$:     MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3639  032160  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3640  032164  006612                    MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3641  032166  112737  000006  172310    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3642  032174  011201                    MOV     (R2),R1         ;READ FROM ADDRESS 60000
3643  032176  020001                    CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
3644  032200  001401                    BEQ     6$              ;BRANCH IF STORE WAS CORRECT
3645  032202  104047                    ERROR   47              ;INCORRECT STORE
3646                                                            ;FOR TIGHTER SCOPE LOOP
3647                                                            ;REPLACE ERROR CALL WITH
3648                                                            ;"BR 5$" = 000765
3649  032204                    6$:     ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3650                                    ;
3651  032204  012737  032230  001110    MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
3652  032212  012737  030340  177776    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3653  032220  012700  125252            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3654  032224  012702  100000            MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3655  032230  010046            8$:     MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3656  032232  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3657  032236  006612                    MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3658  032240  112737  000006  172310    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESISENT
3659  032246  013701  100000            MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3660  032252  020001                    CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3661  032254  001401                    BEQ     9$              ;BRANCH IF STORE WAS CORRECT
3662  032256  104047                    ERROR   47              ;INCORRECT STORE
3663                                                            ;FOR TIGHTER SCOPE LOOP
3664                                                            ;REPLACE ERROR CALL WITH
3665                                                            ;"BR 8$" = 000764
3666  032260                    9$:     ;THIS WILL TEST DSTM = 3 MTPI.
3667  032260  012737  032300  001110    MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3668  032266  012737  030340  177776    MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3669  032274  012700  052525            MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3670  032300  010046            10$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3671  032302  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3672  032306  006637  100000            MTPI    @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
3673  032312  112737  000006  172310    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3674  032320  013701  100000            MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3675  032324  020001                    CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
```

```
3676  032326  001401                          BEQ     11$             ;BRANCH IF STORE WAS CORRECT
3677  032330  104047                          ERROR   47              ;INCORRECT STORE
3678                                                                  ;FOR TIGHTER SCOPE LOOP
3679                                                                  ;REPLACE ERROR CALL WITH
3680                                                                  ;"BR 10$" = 000763
3681  032332                          11$:    ;THIS WILL TEST DSTM = 4 MTPI.
3682  032332  012737  032352  001110          MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3683  032340  012737  030340  177776          MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3684  032346  012700  125252                  MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3685  032352  010046                  12$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3686  032354  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
```

```
3687  032360  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3688  032364  006642                  MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
3689  032366  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3690  032374  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3691  032400  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3692  032402  001401                  BEQ     13$             ;BRANCH IF STORE WAS CORRECT
3693  032404  104047                  ERROR   47              ;INCORRECT STORE
3694                                                          ;FOR TIGHTER SCOPE LOOP
3695                                                          ;REPLACE ERROR CALL WITH
3696                                                          ;"BR 12$" = 000762
3697  032406                  13$:    ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3698                                  ;
3699  032406  012737  032440  001110  MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
3700  032414  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3701  032422  012700  052525          MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3702  032426  012702  001204          MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3703  032432  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3704  032440  010046          14$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3705  032442  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3706  032446  006652                  MTPI    @-(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
3707  032450  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3708  032456  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3709  032462  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3710  032464  001401                  BEQ     15$             ;BRANCH IF STORE WAS CORRECT
3711  032466  104047                  ERROR   47              ;INCORRECT STORE
3712                                                          ;FOR TIGHTER SCOPE LOOP
3713                                                          ;REPLACE ERROR CALL WITH
3714                                                          ;"BR 14$" = 000764
3715  032470                  15$:    ;THIS WILL TEST DSTM = 6 MTPI.
3716                                  ;
3717  032470  012737  032512  001110  MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
3718  032476  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3719  032504  012700  052525          MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3720  032510  005002                  CLR     R2              ;MAKE REGISTER 2 ZERO
3721  032512  010046          16$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3722  032514  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3723  032520  006662  100000          MTPI    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
3724  032524  112737  000006  172310  MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3725  032532  013701  100000          MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3726  032536  020001                  CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3727  032540  001401                  BEQ     17$             ;BRANCH IF STORE WAS CORRECT
3728  032542  104047                  ERROR   47              ;INCORRECT STORE
3729                                                          ;FOR TIGHTER SCOPE LOOP
3730                                                          ;REPLACE ERROR CALL WITH
3731                                                          ;"BR 16$" = 000763
3732  032544                  17$:    ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3733                                  ;
3734  032544  012737  032576  001110  MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
3735  032552  012737  030340  177776  MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3736  032560  012700  125252          MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3737  032564  012737  100000  001202  MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
3738                                                          ;INTO LOCATION $TMP2
3739  032572  012702  001202          MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3740  032576  010046          18$:    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3741  032600  105037  172310          CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3742  032604  006672  000000          MTPI    @0(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
```

```
3743  032610  112737  000006  172310          MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3744  032616  013701  100000                  MOV     @#100000,R1     ;READ FROM ADDRESS 60000
3745  032622  020001                          CMP     R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
3746  032624  001401                          BEQ     19$             ;BRANCH IF STORE WAS CORRECT
3747  032626  104047                          ERROR   47              ;INCORRECT STORE
3748                                                                  ;FOR TIGHTER SCOPE LOOP
3749                                                                  ;REPLACE ERROR CALL WITH
3750                                                                  ;"BR 18$" = 000763
3751  032630  012737  032036  001110  19$:    MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3752  032636  012737  002150  000250          MOV     #MGMERR,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3753  032644  000423                          BR      TST45           ;;BRANCH TO NEXT TEST
3754
3755
3756  032646  012637  001266          20$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3757  032652  012637  001270                  MOV     (KSP)+,TRAPPS
3758  032656  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3759  032664  013737  177576  001274          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3760  032672  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3761  032700  104051                          ERROR   51              ;TRIED TO LOAD A N.R. PAGE 4
3762                                                                  ;FOR TIGHTER SCOPE LOOP
3763                                                                  ;REPLACE ERROR CALL WITH
3764                                                                  ;A "NOP" = 000240
3765  032702  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3766  032706  013746  001266                  MOV     TRAPPC,-(KSP)
3767  032712  000002                          RTI                     ;RETURN TO TEST
3768
3769
3770
3771                                  ;;*****************************************************************
3772                                  ;*TEST 45       MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
3773                                  ;*
3774                                  ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
3775                                  ;*      FETCH IS FROM KERNEL SPACE.
3776                                  ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3777                                  ;*
3778                                  ;*
3779                                  ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3780                                  ;*      WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
3781                                  ;*
3782                                  ;;*****************************************************************
3783  032714  000004                  TST45:  SCOPE
3784  032716  012700  077406          1$:     MOV     #77406,R0       ;MAKE ALL USER I-SPACE PAGES RESIDENT
3785                                                                  ;READ/WRITE, LENGTH 200 BLOCKS
3786  032722  012702  000010                  MOV     #10,R2          ;SET LOOP COUNTER TO 8
3787  032726  012701  177600                  MOV     #U1PDR0,R1      ;LOAD ADDRESS OF FIRST PDR IN R1
3788  032732  010021                  2$:     MOV     R0,(R1)+        ;LOAD PDR WITH 77406
3789  032734  077202                          SOB     R2,2$           ;LOOP UNTIL 8 USER PDRS LOADED
3790  032736  012737  032744  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR TO 3$
3791  032744  012737  140340  177776  3$:     MOV     #140340,PSW     ;GO TO USER MODE FOR THIS TEST
3792  032752  012737  077406  172310          MOV     #77406,KIPDR4   ;KERNEL I-SPACE PAGE 4 READ/WRITE
3793  032760  012737  000600  172350          MOV     #600,KIPAR4     ;MAP KERNEL I PAGE 4 TO 12K
3794  032766  012737  000600  177650          MOV     #600,U1PAR4     ;MAP USER I PAGE 4 TO 12K
3795  032774  012700  036514                  MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3796  033000  010037  100000                  MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3797  033004  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3798                                          ;THE FOLLOWING WILL TEST DSTM=0 MFPI
```

```
3799
3800  033010  012737  033412  000250          MOV     #21$,MMVEC      ;SET M.M. VECTOR TO 21$
3801  033016  105037  177610                  CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3802  033022  012737  140340  177776          MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3803  033030  006506                  4$:     MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
3804  033032  022706  000700                  CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT 1$
3805  033036  001407                          BEQ     5$              ;BRANCH IF NOTHING WAS PUSHED
3806  033040  012600                          MOV     (USP)+,R0       ;POP USER STACK INTO R0
3807  033042  012701  001100                  MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
3808  033046  020001                          CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3809  033050  001403                          BEQ     6$              ;BRANCH IF YOU DID
3810  033052  104046                          ERROR   46              ;WRONG THING WAS PUSHED ON STACK
3811                                                                  ;FOR TIGHTER SCOPE LOOP
3812                                                                  ;REPLACE ERROR CALL WITH
3813                                                                  ;"BR 4$" = 000766
3814  033054  000401                          BR      6$              ;BRANCH TO NEXT TRY
3815  033056  104050                  5$:     ERROR   50              ;NOTHING PUSHED ON STACK
3816                                                                  ;FOR TIGHTER SCOPE LOOP
3817                                                                  ;REPLACE ERROR CALL WITH
3818                                                                  ;"BR 4$" = 000764
3819  033060                          6$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3820  033060  012737  033066  001110          MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
3821  033066  012737  140340  177776  7$:     MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3822  033074  012700  036514                  MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
3823  033100  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3824  033104  006512                          MFPI    (R2)            ;READ FROM PHYSICAL 60000
3825  033106  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3826  033110  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3827  033112  001401                          BEQ     8$              ;BRANCH IF CORRECT DATA WAS FETCHED
3828  033114  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3829                                                                  ;FOR TIGHTER SCOPE LOOP
3830                                                                  ;REPLACE ERROR CALL WITH
3831                                                                  ;"BR 7$" = 000764
3832  033116                          8$:     ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3833  033116  012737  033124  001110          MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3834  033124  012737  140340  177776  9$:     MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3835  033132  012702  100000                  MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3836  033136  006522                          MFPI    (R2)+           ;READ FROM PHYSICAL 60000
3837  033140  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3838  033142  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3839  033144  001401                          BEQ     10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3840  033146  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3841                                                                  ;FOR TIGHTER SCOPE LOOP
3842                                                                  ;REPLACE ERROR CALL WITH
3843                                                                  ;"BR 9$" = 000766
3844  033150                          10$:    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3845  033150  012737  033156  001110          MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
3846  033156  012737  140340  177776  11$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3847  033164  006537  100000                  MFPI    @#100000        ;READ FROM PHYSICAL 60000
3848  033170  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3849  033172  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3850  033174  001401                          BEQ     12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3851  033176  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3852                                                                  ;FOR TIGHTER SCOPE LOOP
3853                                                                  ;REPLACE ERROR CALL WITH
3854                                                                  ;"BR 11$" = 000767
```

```
3855  033200                          12$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3856  033200  012737  033206  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3857  033206  012737  140340  177776  13$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE DERNEL PRESENT USER
3858  033214  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3859  033220  006542                          MFPI    -(R2)           ;READ FROM PHYSICAL 60000
3860  033222  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3861  033224  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3862  033226  001401                          BEQ     14$             ;BRANCH IF CORRECT DATA WAS FETCHED
3863  033230  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3864                                                                  ;FOR TIGHTER SCOPE LOOP
3865                                                                  ;REPLACE ERROR CALL WITH
3866                                                                  ;"BR 13$" = 000766
3867  033232                          14$:    ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3868                                          ;
3869  033232  012737  033240  001110          MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
3870  033240  012737  140340  177776  15$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3871  033246  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3872  033254  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3873  033260  006552                          MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
3874  033262  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3875  033264  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3876  033266  001401                          BEQ     16$             ;BRANCH IF CORRECT DATA FETCHED
3877  033270  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3878                                                                  ;FOR TIGHTER SCOPE LOOP
3879                                                                  ;REPLACE ERROR CALL WITH
3880                                                                  ;"BR 15$" = 000763
3881  033272                          16$:    ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3882                                          ;
3883  033272  012737  033300  001110          MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$.
3884  033300  012737  140340  177776  17$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3885  033306  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3886  033310  006562  100000                  MFPI    100000(R2)      ;READ FROM PHYSICAL 60000
3887  033314  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3888  033316  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3889  033320  001401                          BEQ     18$             ;BRANCH IF CORRECT DATA FETCHED
3890  033322  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3891                                                                  ;FOR TIGHTER SCOPE LOOP
3892                                                                  ;REPLACE ERROR CALL WITH
3893                                                                  ;"BR 17$" = 000766
3894  033324                          18$:    ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3895                                          ;
3896  033324  012737  033332  001110          MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
3897  033332  012737  140340  177776  19$:    MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3898  033340  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3899  033346  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3900  033352  006572  000000                  MFPI    @0(R2)          ;READ FROM PHYSICAL 60000
3901  033356  012601                          MOV     (USP)+,R1       ;POP USER STACK INTO R1
3902  033360  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
3903  033362  001401                          BEQ     20$             ;BRANCH IF CORRECT DATA FETCHED
3904  033364  104046                          ERROR   46              ;WRONG DATA WAS FETCHED
3905                                                                  ;FOR TIGHTER SCOPE LOOP
3906                                                                  ;REPLACE ERROR CALL WITH
3907                                                                  ;"BR 19$" = 000762
3908  033366  012737  002150  000250  20$:    MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3909  033374  012737  000340  177776          MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3910  033402  012737  032716  001110          MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
```

```
3911  033410  000423                          BR      TST46           ;;BRANCH TO NEXT TEXT
3912
3913
3914  033412  012637  001266          21$:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3915  033416  012637  001270                  MOV     (KSP)+,TRAPPS
3916  033422  013737  177572  001272          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3917  033430  013737  177576  001274          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3918  033436  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3919  033444  104051                          ERROR   51              ;TRIED TO READ NON-RESIDENT PAGE
3920                                                                  ;FOR TIGHTER SCOPE LOOP
3921                                                                  ;REPLACE ERROR CALL WITH
3922                                                                  ;A "NOP" = 000240
3923  033446  013746  001270                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3924  033452  013746  001266                  MOV     TRAPPC,-(KSP)
3925  033456  000002                          RTI                     ;RETURN TO TEST
3926
3927                                  ;;**********************************************************************
3928                                  ;*TEST 46       MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
3929                                  ;*
3930                                  ;*      THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
3931                                  ;*      BETWEEN INSTRUCTION AND DATA SPACE IN THE FONZ-11
3932                                  ;*      MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI.
3933                                  ;*
3934                                  ;;**********************************************************************
3935  033460  000004                  TST46:  SCOPE
3936  033462  012737  030340  177776  1$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
3937  033470  106506                          MFPD    USP             ;MFPD SHOULD ACT LIKE MFPI PUTTING
3938                                                                  ;USER STACK POINTER ON THE KERNEL STACK
3939  033472  022706  001100                  CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON KERNEL STACK?
3940  033476  001407                          BEQ     2$              ;BRANCH IF NO
3941  033500  012600                          MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3942  033502  012701  000700                  MOV     #USESTK,R1      ;EXPECTING TO GET 700 AS USP
3943  033506  020001                          CMP     R0,R1           ;DID GET RIGHT POINTER VALUE?
3944  033510  001403                          BEQ     3$              ;BRANCH IF YES
3945  033512  104053                          ERROR   53              ;WRONG THING WAS PUSHED ON STACK
3946                                                                  ;FOR TIGHTER SCOPE LOOP
3947                                                                  ;REPLACE ERROR CALL WITH
3948                                                                  ;"BR 1$" = 000763
3949  033514  000401                          BR      3$              ;BRANCH TO NEXT TRY
3950  033516  104054                  2$:     ERROR   54              ;NOTHING PUSHED ON STACK
3951                                                                  ;FOR TIGHTER SCOPE LOOP
3952                                                                  ;REPLACE ERROR CALL WITH
3953                                                                  ;"BR 1$" = 000761
3954  033520  012737  033526  001110  3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3955  033526  012746  007777          4$:     MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3956  033532  106606                          MTPD    USP             ;LOAD THE USER STACK POINTER
3957  033534  106506                          MFPD    USP             ;READ USER STACK POINTER
3958  033536  012601                          MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3959  033540  022701  007777                  CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED?
3960  033544  001401                          BEQ     5$              ;BRANCH IF YES
3961  033546  104054                          ERROR   54              ;USER STACK POINTER NOT CHANGED
3962                                                                  ;FOR TIGHTER SCOPE LOOP
3963                                                                  ;REPLACE ERROR CALL WITH
3964                                                                  ;"BR 4$" = 000767
3965  033550  012746  000700          5$:     MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK. PTR.
3966  033554  106606                          MTPD    USP             ;RESTORE USER STACK POINTER
```

```
                                        MOV     #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST

                                ;;*****************************************************************
                                ;*TEST 47        MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
                                ;*
                                ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
                                ;*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
                                ;*      STACK IS NOT DECREMENTED BEFORE ACCESS.
                                ;*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
                                ;*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
                                ;;*****************************************************************
033564  000004                  TST47:  SCOPE
033566  005037  177776          1$:     CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
033572  012700  001100                  MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
033576  010006                          MOV     R0,KSP          ;LOAD STACK POINTER
033600  006506                          MFPI    KSP             ;THE VALUE "STACK" SHOULD BE PUSHED
                                                                ;BEFORE BEING DECREMENTED
033602  011601                          MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
033604  020001                          CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE
                                                                ;STACK POINTER PUSHED?
033606  001401                          BEQ     2$              ;BRANCH IF YES
033610  104046                          ERROR   46              ;MFPI FETCHED WRONG DATA
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 1$" = 000766
033612  005740                  2$:     TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
033614  020600                          CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
033616  001401                          BEQ     3$              ;BRANCH IF YES
033620  104050                          ERROP   50              ;STACK NOT PUSHED BY THE MFPI
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;"BR 1$" = 000762
033622  012706  001100          3$:     MOV     #STACK,KSP      ;RESTORE STACK POINTER


                                .SBTTL  ***************************************************************
```

```
033626  013746  000004          TIMOFF: MOV     @#ERRVEC,-(SP)  ;SAVE CONTENTS OF LOC 4
033632  012737  033644  000004          MOV     #1$,@#ERRVEC    ;SETUP IN CASE OF TIMEOUT
033640  005337  164000                  DEC     @#164000        ;TURN OFF TIMER ON MULTI-TESTER
033644  012637  000004          1$:     MOV     (SP)+,@#ERRVEC  ;RESTORE CONTENTS OF LOC 4

                                .SBTTL  END OF PASS ROUTINE


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

033650                          $EOP:
033650  000004                          SCOPE
033652  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
033656  005037  001212                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
033662  005237  001234                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
033666  042737  100000  001234          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
033674  005327                          DEC     (PC)+           ;;LOOP?
033676  000001                  $EOPCT: .WORD   1
033700  003072                          BGT     $DOAGN          ;;YES
033702  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
033704  000001                  $ENDCT: .WORD   1
033706  033676                          $EOPCT
033710  104401  033716                  TYPE    ,65$            ;;TYPE ASCIZ STRING
033714  000407                          BR      64$             ;;GET OVER THE ASCIZ
033716  006412  047105  020104  65$:    .ASCIZ  <12><15>/END PASS #/
033724  040520  051523  021440
033732     000
        033734                  .EVEN
033734                          64$:
033734  013746  001234                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                ;;TYPE PASS NUMBER
033740  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCIZ WITH SIGN
033742  104401  033750                  TYPE    ,67$            ;;TYPE ASCII STRING
033746  000421                          BR      66$             ;;GET OVER THE ASCIZ
033750  035411  047524  040524  67$:    .ASCIZ  /       ;TOTAL ERRORS SINCE LAST REPORT /
033756  020114  051105  047522
033764  051522  051440  047111
033772  042503  046040  051501
034000  020124  042522  047520
034006  052122  000040
                                .EVEN
034012                          66$:
034012  013746  001112                  MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                ;;TOTAL NUMBER OF ERRORS
034016  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
034020  104401  001223                  TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
034024  005037  001112                  CLR     $ERTTL          ;;CLEAR ERROR TOTAL
034030  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
034034  001414                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
034036  005046                          CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
034040  012746  034046                  MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
```

```
4063  034044  000426                          BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
4064                                                                  ;;WITH A CLEARED "T" BIT
4065  034046                          $CLR.T:
4066  034046  013700  000042                  MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
4067  034052  001405                          BEQ     $DOAGN          ;;RETURN ADDRESS
4068  034054  000005                          RESET                   ;;CLEAR THE WORLD
4069  034056  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
4070  034060  000240                          NOP                     ;;SAVE ROOM
4071  034062  000240                          NOP                     ;;FOR
4072  034064  000240                          NOP                     ;;ACT11
4073
4074  034066                          $DOAGN:
4075  034066  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
4076  034070  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE "T" BIT
4077  034074  032777  010000  145036          BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
4078  034102  001005                          BNE     1$              ;;BR IF NO
4079  034104  005137  034130                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
4080  034110  100402                          BMI     1$              ;;BR IF NO
4081  034112  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
4082  034116  012746  034124          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
4083  034122  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
4084                                                                  ;;AN "RTT" IF "RTT" IS A LEGAL
4085                                                                  ;;INSTRUCTION
4086  034124                          $LOOP:
4087  034124  000137                          JMP     @(PC)+          ;;RETURN
4088  034126  020462                  $RTNAD: .WORD   RESTRT
4089  034130  000000                  $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
4090  034132     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
4091           034136                         .EVEN
4092                                  .SBTTL  SCOPE HANDLER ROUTINE
4093
4094                                  ;;**************************************************************************
4095                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4096                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4097                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4098                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4099                                  ;*SW14=1         LOOP ON TEST
4100                                  ;*SW11=1         INHIBIT ITERATIONS
4101                                  ;*SW09=1         LOOP ON ERROR
4102                                  ;*SW08=1         LOOP ON TEST IN SWR<7:0>
4103                                  ;*CALL
4104                                  ;*      SCOPE           ;;SCOPE=IOT
4105
4106  034136                          $SCOPE:
4107  034136  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
4108  034140  032777  040000  144772  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
4109  034146  001114                          BNE     $OVER           ;;YES IF SW14=1
4110                                  ;#####START OF CODE FOR THE XOR TESTER#####
4111  034150  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4112                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
4113  034152  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4114  034156  012737  034176  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
4115  034164  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
4116  034170  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4117  034174  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
4118  034176  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
```

```
 4119  034200  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 4120  034204  000423                         BR      7$              ;;LOOP ON THE PRESENT TEST
 4121  034206                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 4122  034206  032777  000400  144724         BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 4123  034214  001404                         BEQ     2$              ;;BR IF NO
 4124  034216  127737  144716  001102         CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
 4125  034224  001465                         BEQ     $OVER           ;;BR IF YES
 4126  034226  105737  001103          2$:    TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 4127  034232  001421                         BEQ     3$              ;;BR IF NO
 4128  034234  123737  001115  001103         CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 4129  034242  101015                         BHI     3$              ;;BR IF NO
 4130  034244  032777  001000  144666         BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 4131  034252  001404                         BEQ     4$              ;;BR IF NO
 4132  034254  013737  001110  001106  7$:    MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 4133  034262  000446                         BR      $OVER
 4134  034264  105037  001103          4$:    CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 4135  034270  005037  001212                 CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 4136  034274  000415                         BR      1$              ;;ESCAPE TO THE NEXT TEST
 4137  034276  032777  004000  144634  3$:    BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 4138  034304  001011                         BNE     1$              ;;BR IF YES
 4139  034306  005737  001234                 TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 4140  034312  001406                         BEQ     1$              ;;      INHIBIT ITERATIONS
 4141  034314  005237  001104                 INC     $ICNT           ;;INCREMENT ITERATION COUNT
 4142  034320  023737  001212  001104         CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 4143  034326  002024                         BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 4144  034330  012737  000001  001104  1$:    MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 4145  034336  013737  034414  001212         MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 4146  034344  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 4147  034350  113737  001102  001232         MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 4148  034356  011637  001106                 MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 4149  034362  011637  001110                 MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 4150  034366  005037  001214                 CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 4151  034372  112737  000001  001115         MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 4152  034400  013777  001102  144534  $OVER: MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 4153  034406  013716  001106                 MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 4154  034412  000002                         RTI                     ;;FIXES PS
 4155  034414  000200                 $MXCNT: 200                     ;;MAX. NUMBER OF ITERATIONS
 4156                                  .SBTTL  ERROR HANDLER ROUTINE
 4157
 4158                                  ;;**************************************************************
 4159                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 4160                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 4161                                  ;*AND GO TO ERRTYP ON ERROR
 4162                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 4163                                  ;*SW15=1         HALT ON ERROR
 4164                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
 4165                                  ;*SW10=1         BELL ON ERROR
 4166                                  ;*SW09=1         LOOP ON ERROR
 4167                                  ;*CALL
 4168                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 4169
 4170  034416                          $ERROR:
 4171  034416  104410                         CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 4172  034420  010037  001162                 MOV     R0,$REG0        ;SAVE THE CONTENTS OF R0
 4173  034424  010137  001164                 MOV     R1,$REG1        ;SAVE THE CONTENTS OF R1
 4174  034430  010237  001166                 MOV     R2,$REG2        ;SAVE THE CONTENTS OF R2
```

```
4175  034434  010337  001170                  MOV     R3,$REG3        ;SAVE THE CONTENTS OF R3
4176  034440  010437  001172                  MOV     R4,$REG4        ;SAVE THE CONTENTS OF R4
4177  034444  010537  001174                  MOV     R5,$REG5        ;SAVE THE CONTENTS OF R5
4178  034450  113737  001102  001262          MOVB    $TSTNM,TESTNO   ;SAVE THE TEST NUMBER
4179  034456  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
4180  034462  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
4181  034464  013777  001102  144450          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4182  034472  032777  002000  144440          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
4183  034500  001402                          BEQ     1$              ;;NO - SKIP
4184  034502  104401  001216                  TYPE    ,$BELL          ;;RING BELL
4185  034506  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
4186  034512  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
4187  034516  162737  000002  001116          SUB     #2,$ERRPC
4188  034524  117737  144366  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
4189  034532  032777  020000  144400          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
4190  034540  001004                          BNE     20$             ;;SKIP TYPEOUTS
4191  034542  004737  034654                  JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
4192  034546  104401  001223                  TYPE    ,$CRLF
4193  034552                          20$:
4194  034552  122737  000001  001246          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4195  034560  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
4196  034562  113737  001114  034574          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
4197  034570  004737  037206                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
4198  034574     000          21$:     .BYTE   0
4199  034575     000                  .BYTE   0
4200  034576  000777          22$:    BR      22$             ;;APT ERROR LOOP
4201  034600  005777  144334          2$:     TST     @SWR            ;;HALT ON ERROR
4202  034604  100002                          BPL     3$              ;;SKIP IF CONTINUE
4203  034606  000000                          HALT                    ;;HALT ON ERROR!
4204  034610  104410                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
4205  034612  032777  001000  144320  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
4206  034620  001402                          BEQ     4$              ;;BR IF NO
4207  034622  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
4208  034626  005737  001214          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
4209  034632  001402                          BEQ     5$              ;;BR IF NONE
4210  034634  013716  001214                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
4211  034640                          5$:
4212  034640  022737  034056  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
4213  034646  001001                          BNE     6$              ;;BRANCH IF NO
4214  034650  000000                          HALT                    ;;YES
4215  034652                          6$:
4216  034652  000002                          RTI                     ;;RETURN
4217                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4218
4219                                  ;;**********************************************************************
4220                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4221                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4222                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4223                                  ;*
4224                                  ;*NOTES:
4225                                  ;*1)    THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
4226                                  ;*      FOR "EM", "DH", AND "DT".
4227                                  ;*2)    TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
4228                                  ;*3)    FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4229                                  ;*4)    THE AVAILABLE FORMATS FOR TYPING DATA ARE:
4230                                  ;*      DF      FORMAT
```

```
                        ;*      0       TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
                        ;*      1       TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
                        ;*      2       TYPE A 16 DIGIT BINARY NUMBER
                        ;*      3       TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
                        ;*

034654                  ERRTYP:
034654  104401  001223          TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
034660  010046                  MOV     R0,-(SP)        ;SAVE R0
034662  005000                  CLR     R0              ;PICKUP THE ITEM INDEX
034664  153700  001114          BISB    @#$ITEMB,R0
034670  001004                  BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
                                                        ;TYPE THE PC OF THE ERROR
034672  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                        ;;ERROR ADDRESS
034676  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034700  000471                  BR      13$             ;GET OUT
034702  005300          1$:     DEC     R0              ;ADJUST THE INDEX SO THAT IT WILL
034704  006300                  ASL     R0              ;       WORK FOR THE ERROR TABLE
034706  006300                  ASL     R0
034710  006300                  ASL     R0
034712  062700  001316          ADD     #$ERRTB,R0      ;FORM TABLE POINTER
034716  012037  034726          MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
034722  001404                  BEQ     3$              ;SKIP TYPEOUT IF NO POINTER
034724  104401                  TYPE                    ;TYPE THE "ERROR MESSAGE"
034726  000000          2$:     .WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
034730  104401  001223          TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
034734  012037  034744  3$:     MOV     (R0)+,4$        ;PICKUP "DATA HEADER" POINTER
034740  001404                  BEQ     5$              ;SKIP TYPEOUT IF 0
034742  104401                  TYPE                    ;TYPE THE "DATA HEADER"
034744  000000          4$:     .WORD   0               ;"DATA HEADER" POINTER GOES HERE
034746  104401  001223          TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
034752  010146          5$:     MOV     R1,-(SP)        ;SAVE R1
034754  012001                  MOV     (R0)+,R1        ;PICKUP "DATA TABLE" POINTER
034756  001441                  BEQ     12$             ;BR IF NO DATA TO BE TYPED
034760  012000                  MOV     (R0)+,R0        ;PICKUP "DATA FORMAT" POINTER
034762  105710          6$:     TSTB    (R0)            ;IS IT FORMAT 0?
034764  001003                  BNE     7$              ;BR IF NO

                        ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
034766  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
034770  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034772  000425                  BR      11$

                        ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
034774  121027  000001  7$:     CMPB    (R0),#1         ;IS IT FORMAT 1?
035000  001003                  BNE     8$              ;BRANCH IF NO
035002  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
035004  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
035006  000417                  BR      11$

                        ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
035010  121027  000002  8$:     CMPB    (R0),#2         ;IS IT FORMAT 2?
035014  001003                  BNE     9$              ;BRANCH IF NO
035016  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
035020  104406                  TYPBN                   ;;GO TYPE--BINARY ASCII
```

```
 4287  035022  000411                        BR      11$
 4288
 4289                                        ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
 4290  035024  012146                        9$:     MOV     (R1)+,-(SP)     ;PUT ADDRESS OF FIRST LOC. ON STACK
 4291  035026  004737  040260                        JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
 4292  035032  062716  000005                        ADD     #5,(SP)         ;ONLY NEED 6 CHARACTERS NOT 11
 4293  035036  012637  035044                        MOV     (SP)+,10$       ;PUT ADDRESS OF ASCII CHARS. AT 10$
 4294  035042  104401                              TYPE                    ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.
 4295  035044  000000                        10$:    .WORD   0
 4296
 4297  035046  005711                        11$:    TST     (R1)            ;IS THERE ANOTHER NUMBER?
 4298  035050  001404                              BEQ     12$             ;BR IF NO
 4299  035052  104401  035074                        TYPE    ,14$            ;TYPE TWO(2) SPACES
 4300  035056  105720                              TSTB    (R0)+           ;POINT TO NEW "DATA FORMAT"
 4301  035060  000740                              BR      6$              ;LOOP
 4302
 4303  035062  012601                        12$:    MOV     (SP)+,R1        ;RESTORE R1
 4304  035064  012600                        13$:    MOV     (SP)+,R0        ;RESTORE R0
 4305  035066  104401  001223                        TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
 4306  035072  000207                              RTS     PC              ;RETURN
 4307  035074  020040      000               14$:    .ASCIZ  /  /            ;TWO(2) SPACES
 4308          035100                              .EVEN
 4309
```

```
4310                                    .SBTTL  ***** SUBROUTINES USED BY THIS PROGRAM *****
4311
4312                                    .SBTTL  TURN OFF T-BIT AND SAVE CURRENT PSW
4313                                    ;;*****************************************************************
4314                                    ;*
4315                                    ;*      THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
4316                                    ;*      IF IT IS ON.  THE PROCESSOR STATUS IS SAVED IN "TBITPS" SO THAT
4317                                    ;*      THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
4318                                    ;*      WARRANT T-BIT TRAPPING.
4319                                    ;*
4320                                    ;;*****************************************************************
4321  035100  033727  177776  000020  TOFF:   BIT     PSW,#TBIT       ;IS THE T-BIT SET IN THE PSW?
4322  035106  001411                          BEQ     1$              ;EXIT IF NO
4323  035110  013746  177776                  MOV     PSW,-(SP)       ;PUSH PRESENT PSW ON THE STACK
4324  035114  011637  001276                  MOV     (SP),TBITPS     ;ALSO SAVE IT IN "TBITPS" FOR
4325                                                                  ;RESTORING LATER
4326  035120  042716  000020                  BIC     #TBIT,(SP)      ;CLEAR THE T-BIT (BIT 4) IN THE PSW
4327  035124  012746  035132                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
4328  035130  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT OFF
4329  035132  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM
4330
4331                                    .SBTTL  TURN ON T-BIT AND RESTORE PREVIOUS PSW
4332                                    ;;*****************************************************************
4333                                    ;*
4334                                    ;*      THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
4335                                    ;*      PREVIOUS CONDITION BY RESTORING THE "T-BIT PSW" SAVED BY THE
4336                                    ;*      "TOFF" SUBROUTINE IN THE "TBITPS" LOCATION.
4337                                    ;*
4338                                    ;;*****************************************************************
4339  035134  033727  001276  000020  TON:    BIT     TBITPS,#TBIT    ;WAS T-BIT ON IN THE PREVIOUS PSW?
4340  035142  001410                          BEQ     1$              ;EXIT IF NO
4341  035144  013746  001276                  MOV     TBITPS,-(SP)    ;PUSH PREVIOUS PSW ON THE STACK
4342  035150  012737  000340  001276          MOV     #340,TBITPS     ;RESET THE "TBITPS" LOCATION
4343  035156  012746  035164                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
4344  035162  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT RESTORED
4345  035164  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM
4346
4347                                    .SBTTL  SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4348                                    ;;*****************************************************************
4349                                    ;*
4350                                    ;*      THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
4351                                    ;*      TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
4352                                    ;*      PDR'S TO A 1.  THE "INITIAL STATE" OF HAVING ALL BITS=1 IS
4353                                    ;*      USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
4354                                    ;*      A SINGLE PAR OR PDR ADDRESS.
4355                                    ;*
4356                                    ;;*****************************************************************
4357
4358  035166  012702  000010  SETREG: MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
4359  035172  012701  172300          MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
4360  035176  012721  177777  1$:     MOV     #-1,(R1)+       ;SET BITS IN KERNEL PDR TO 1
4361  035202  077203                  SOB     R2,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
4362  035204  012702  000010          MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
4363  035210  012701  172340          MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
4364  035214  012721  177777  2$:     MOV     #-1,(R1)+       ;SET BITS IN A KERNEL PAR TO 1
4365  035220  077203                  SOB     R2,2$           ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

```
4366  035222  012702  000010              MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
4367  035226  012701  177600              MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST PDR INTO R1
4368  035232  012721  177777      3$:     MOV     #-1,(R1)+       ;SET BITS IN A USER PDR TO 1
4369  035236  077203                      SOB     R2,3$           ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
4370  035240  012702  000010              MOV     #10,R2          ;LOAD LOOP COUNTER WITH AN 8
4371  035244  012701  177640              MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST PAR INTO R1
4372  035250  012721  177777      4$:     MOV     #-1,(R1)+       ;SET BITS IN A USER PAR TO 1
4373  035254  077203                      SOB     R2,4$           ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
4374  035256  000207                      RTS     PC              ;RETURN TO TEST
4375
4376                              .SBTTL  READ & COMPARE KERNEL & USER PAR/PDR'S
4377                              ;;*****************************************************************
4378                              ;*
4379                              ;*      THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
4380                              ;*      READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
4381                              ;*      WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
4382                              ;*      ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
4383                              ;*      BE REPORTED BY THIS SUBROUTINE.
4384                              ;*
4385                              ;;*****************************************************************
4386  035260                      CMPREG:
4387  035260  012701  172300              MOV     #KIPDR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
4388  035264  012704  000010              MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4389  035270  012705  077416              MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
4390  035274  021105              1$:     CMP     (R1),R5         ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4391  035276  001404                      BEQ     2$              ;BRANCH IF YES
4392  035300  020100                      CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4393  035302  001402                      BEQ     2$              ;BRANCH IF YES
4394  035304  011102                      MOV     (R1),R2         ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4395  035306  104016                      ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4396                                                              ;FOR TIGHTER SCOPE LOOP
4397                                                              ;REPLACE ERROR CALL WITH
4398                                                              ;AN "RTS PC" = 000207
4399  035310  062701  000002      2$:     ADD     #2,R1           ;FORM NEXT ADDRESS
4400  035314  077411                      SOB     R4,1$           ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
4401  035316  012701  172340              MOV     #KIPAR0,R1      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
4402  035322  012704  000010              MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4403  035326  012705  177777              MOV     #177777,R5      ;PUT EXPECTED PAR CONTENTS IN R5
4404  035332  021105              3$:     CMP     (R1),R5         ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4405  035334  001404                      BEQ     4$              ;BRANCH IF YES
4406  035336  020100                      CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4407  035340  001402                      BEQ     4$              ;BRANCH IF YES
4408  035342  011102                      MOV     (R1),R2         ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4409  035344  104016                      ERROR   16              ;A PAR WAS EFFECTED BY CLEARING A DIFFENENT PAR/PDR
4410                                                              ;FOR TIGHTER SCOPE LOOP
4411                                                              ;REPLACE ERROR CALL WITH
4412                                                              ;"RTS PC" = 000207
4413  035346  062701  000002      4$:     ADD     #2,R1           ;NEXT ADDRESS
4414  035352  077411                      SOB     R4,3$           ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
4415  035354  012701  177600              MOV     #UIPDR0,R1      ;LOAD ADDRESS OF FIRST USER PDR IN R1
4416  035360  012704  000010              MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4417  035364  012705  077416              MOV     #77416,R5       ;PUT EXPECTED PDR CONTENTS IN R5
4418  035370  021105              5$:     CMP     (R1),R5         ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4419  035372  001404                      BEQ     6$              ;BRANCH IF YES
4420  035374  020100                      CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4421  035376  001402                      BEQ     6$              ;BRANCH IF YES
```

```
4422  035400  011102                    MOV     (R1),R2         ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4423  035402  104016                    ERROR   16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4424                                                            ;FOR TIGHTER SCOPE LOOP
4425                                                            ;REPLACE ERROR CALL WITH
4426                                                            ;AN "RTS PC" = 000207
4427  035404  062701  000002          6$:     ADD     #2,R1           ;FORM NEXT ADDRESS
4428  035410  077411                          SOB     R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4429  035412  012701  177640                  MOV     #UIPAR0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
4430  035416  012704  000010                  MOV     #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4431  035422  012705  177777                  MOV     #177777,R5      ;PUT EXPECTED PAR CONTENTS IN R5
4432  035426  021105                  7$:     CMP     (R1),R5         ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4433  035430  001404                          BEQ     8$              ;BRANCH IF YES
4434  035432  020100                          CMP     R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4435  035434  001402                          BEQ     8$              ;BRANCH IF YES
4436  035436  011102                          MOV     (R1),R2         ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4437  035440  104016                          ERROR   16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4438                                                            ;FOR TIGHTER SCOPE LOOP
4439                                                            ;REPLACE ERROR CALL WITH
4440                                                            ;AN "RTS PC" = 000207
4441  035442  062701  000002          8$:     ADD     #2,R1           ;FORM NEXT ADDRESS
4442  035446  077411                          SOB     R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4443  035450  000207                          RTS     PC              ;RETURN TO TEST
4444
4445                                  .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
4446                                  ;;**************************************************************
4447                                  ;*
4448                                  ;*      THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
4449                                  ;*      (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
4450                                  ;*      PAGE ADDRESS REGISTER (PAR).  THE SAME METHOD USED BY THE MEMORY
4451                                  ;*      MANAGEMENT LOGIC IS USED.  VBA <15:13> SELECTS WHICH PAR/PDR
4452                                  ;*      IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
4453                                  ;*      TO PAR <11:00> TO GIVE PBA <17:6>.  BITS <17:16> OF THE
4454                                  ;*      PHYSICAL ADDRESS ARE LEFT IN LOC. "PBAHI" AND BITS <15:00>
4455                                  ;*      ARE LEFT IN LOC. "PBALO".  THE PSW'S "CURRENT MODE" BITS
4456                                  ;*      ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S.  THE ROUTINE
4457                                  ;*      IS ENTERED WITH LOC. "VIRT1" CONTAINING THE 16-BIT VIRTUAL
4458                                  ;*      ADDRESS.
4459                                  ;*
4460                                  ;;**************************************************************
4461
4462  035452  012702  172340          FORMPA: MOV     #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4463  035456  032737  140000  177776          BIT     #140000,PSW     ;IN USER MODE?
4464  035464  001402                          BEQ     1$              ;BRANCH IF NO
4465  035466  012702  177640                  MOV     #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
4466  035472  013700  001306          1$:     MOV     VIRT1,R0        ;LOAD VIRTUAL ADDR. (VBA) INTO R0
4467  035476  072027  177764                  ASH     #-14,R0         ;GET BITS <15:13> DOWN TO BITS <3:1>
4468  035502  042700  177761                  BIC     #177761,R0      ;MASK OF ALL BITS BUT BITS <3:1>
4469  035506  060002                          ADD     R0,R2           ;ADD OFFSET TO BASE PAR ADDRESS
4470  035510  011200                          MOV     (R2),R0         ;GET BITS <11:00> FROM APPROPRIATE PAR
4471  035512  010002                          MOV     R0,R2           ;COPY PAR BITS <11:00> INTO R2
4472  035514  013737  001306  001312          MOV     VIRT1,PBALO     ;PUT VIRTUAL ADDR. IN LOC. "PBALO"
4473  035522  042737  160000  001312          BIC     #160000,PBALO   ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4474  035530  072227  177766                  ASH     #-12,R2         ;GET PAR <11:00> DOWN TO BITS <1:0> OF R2
4475  035534  042702  177774                  BIC     #177774,R2      ;CLEAR OFF ALL BITS BUT BITS <1:0>
4476  035540  072027  000006                  ASH     #6,R0           ;SHIFT PAR<9:0> TO <15:6> OF R0
4477  035544  042700  000077                  BIC     #77,R0          ;CLEAR BITS <5:0> OF R0
```

```
 4478  035550  060037  001312          ADD     R0,PBALO        ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
 4479                                                          ;(PAR<9:0> IN BITS <15:6> OF R0)
 4480  035554  005502                  ADC     R2              ;ADD ANY CARRY TO R2
 4481  035556  010237  001314          MOV     R2,PBAHI        ;PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
 4482  035562  000207                  RTS     PC              ;RETURN TO PROGRAM
 4483
 4484
```

```
                                        .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.
035564  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
035572  001114                          BNE     15$             ;;BRANCH IF NO
035574  105777  143344                  TSTB    @$TKS           ;;CHAR THERE?
035600  100111                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
035602  117746  143340                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
035606  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
035612  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
035616  001102                          BNE     15$             ;;NO, RETURN TO USER
035620  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
035626  001476                          BEQ     15$             ;;BRANCH IF YES
                                
035630  104401  036526                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
035634  104401  036533          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
035640  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
035644  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035646  104401  036544                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
035652  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
035654  005046                          CLR     -(SP)           ;;THE NEW SWR
035656  105777  143262          7$:     TSTB    @$TKS           ;;CHAR THERE?
035662  100375                          BPL     7$              ;;IF NOT TRY AGAIN
                                
035664  117746  143256                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
035670  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
                                
035674  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
035700  001015                          BNE     9$              ;;BRANCH IF NOT
035702  104401  036514                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
035706  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
035712  123727  001135  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
035720  001003                          BNE     8$              ;;BRANCH IF NO
035722  012777  000100  143214          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
035730  000137  036556          8$:     JMP     CNTRLC          ;;CONTROL-C RESTART
                                
035734  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
035740  001005                          BNE     10$             ;;BRANCH IF NOT
035742  104401  036521                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
035746  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
035752  000737                          BR      19$             ;;LET'S TRY IT AGAIN
                                
035754  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
035760  001022                          BNE     16$             ;;BRANCH IF NO
035762  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
035766  001403                          BEQ     11$             ;;BRANCH IF YES
035770  016677  000002  143142          MOV     2(SP),@SWR      ;;SAVE NEW SWR
```

```
4541  035776  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
4542  036002  104401  001223          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
4543  036006  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
4544  036014  001003                          BNE     15$             ;;BRANCH IF NOT
4545  036016  012777  000100  143120          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
4546  036024  000002                  15$:    RTI                     ;;RETURN
4547  036026  004737  037120          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
4548  036032  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
4549  036036  002420                          BLT     18$             ;;BRANCH IF YES
4550  036040  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
4551  036044  003015                          BGT     18$             ;;BRANCH IF YES
4552  036046  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
4553  036052  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
4554  036056  001403                          BEQ     17$             ;;BRANCH IF YES
4555  036060  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
4556  036062  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
4557  036064  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
4558  036066  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
4559  036072  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
4560  036076  000667                          BR      7$              ;;GET THE NEXT ONE
4561  036100  104401  001222          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
4562  036104  000720                          BR      20$             ;;SIMULATE CONTROL-U
4563                                  .DSABL  LSB
4564
4565
4566                                  ;;*****************************************************************
4567                                  ;*THIS ROU INE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4568                                  ;*CALL:
4569                                  ;*      RDCHP                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4570                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4571                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
4572                                  ;
4573
4574  036106  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4575  036110  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
4576  036116  105777  143022          1$:     TSTB    @$TKS           ;;WAIT FOR
4577  036122  100375                          BPL     1$              ;;A CHARACTER
4578  036124  117766  143016  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
4579  036132  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4580  036140  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
4581  036146  001013                          BNE     3$              ;;BRANCH IF NO
4582  036150  105777  142770          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
4583  036154  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
4584  036156  117746  142764                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
4585  036162  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
4586  036166  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
4587  036172  001366                          BNE     2$              ;;IF NOT DISCARD IT
4588  036174  000750                          BR      1$              ;;YES, RESUME
4589  036176  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
4590  036204  002407                          BLT     4$              ;;BRANCH IF YES
4591  036206  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
4592  036214  003003                          BGT     4$              ;;BRANCH IF YES
4593  036216  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
4594  036224  000002                  4$:     RTI                     ;;GO BACK TO USER
4595                                  ;;*****************************************************************
4596                                  ;*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

036226  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
036230  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
036232  012703  036504          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
036236  022703  036514          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
036242  101467                          BLOS    4$              ;;BR IF YES
036244  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
036246  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
036250  122713  000003                  CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
036254  001006                          BNE     10$             ;;BRANCH IF NO
036256  104401  036514                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
036262  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
036264  012603                          MOV     (SP)+,R3        ;;RESTORE R3
036266  000137  036556                  JMP     CNTRLC          ;;GOTO CONTROL-C RESTART
036272  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
036276  001022                          BNE     5$              ;;BR IF NO
036300  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
036302  001007                          BNE     6$              ;;BR IF NO
036304  112737  000134  036502        MOVB    #'\,9$          ;;TYPE A BACK SLASH
036312  104401  036502                  TYPE    ,9$
036316  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
036322  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
036324  020327  036504                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
036330  103434                          BLO     4$              ;;BR IF YES
036332  111337  036502                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
036336  104401  036502                  TYPE    ,9$             ;;GO TYPE
036342  000735                          BR      2$              ;;GO READ ANOTHER CHAR.
036344  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
036346  001406                          BEQ     7$              ;;BR IF NO
036350  112737  000134  036502        MOVB    #'\,9$          ;;TYPE A BACK SLASH
036356  104401  036502                  TYPE    ,9$
036362  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
036364  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
036370  001003                          BNE     8$              ;;BR IF NO
036372  104401  036521                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
036376  000715                          BR      1$              ;;GO START OVER
036400  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
036404  001011                          BNE     3$              ;;BRANCH IF NO
036406  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
036410  104401  001223                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
036414  104401  036504                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
036420  000706                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
036422  104401  001222          4$:     TYPE    ,$QUES          ;;TYPE A '?'
036426  000701                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
036430  111337  036502          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
036434  104401  036502                  TYPE    ,9$
036440  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
036444  001274                          BNE     2$              ;;LOOP IF NOT RETURN
036446  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
036452  104401  001224                  TYPE    ,$LF            ;;TYPE A LINE FEED
036456  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
036460  012603                          MOV     (SP)+,R3        ;;RESTORE R3
```

```
 4653  036462  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
 4654  036464  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
 4655  036472  012766  036504  000004        MOV     #$TTYIN,4(SP)
 4656  036500  000002                        RTI                     ;;RETURN
 4657  036502     000                  9$:    .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
 4658  036503     000                         .BYTE   0               ;;TERMINATOR
 4659  036504  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
 4660  036514  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
 4661  036521     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
 4662  036526  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
 4663  036533     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
 4664  036540  036440  000040
 4665  036544  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
 4666  036552  020075     000
 4667          036556                  .EVEN
 4668
 4669                                  .SBTTL  CONTROL-C SERVICING ROUTINE
 4670
 4671                                  ;*      THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
 4672                                  ;*      BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
```

```
4673                                    ;*      (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
4674                                    ;*      A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
4675                                    ;*      THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
4676                                    ;*      AND THEN THE PROGRAM WILL GO TO "END-OF-PASS" AND CONTINUE
4677
4678  036556  013737  001234  001210  CNTRLC: MOV     $PASS,$TMP5     ;GET THE VALUE OF "$PASS"
4679  036564  005237  001210          CNTRLC: INC     $TMP5           ;FORM CURRENT PASS NO.
4680  036570  104401  036635          CNTRLC: TYPE    ,CMSG           ;TYPE THE TEST STOPS MESSAGE
4681  036574  113737  001102  036630 CNTRLC: MOVB    $TSTNM,1$       ;SAVE THE TEST NUMBER
4682  036602  013746  036630          CNTRLC: MOV     1$,-(SP)        ;;SAVE 1$ FOR TYPEOUT
4683  036606  104402                  CNTRLC: TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4684  036610  104401  036632          CNTRLC: TYPE    ,2$             ;TYPE 2 SPACES
4685  036614  013746  001210          CNTRLC: MOV     $TMP5,-(SP)     ;;SAVE $TMP5 FOR TYPEOUT
4686  036620  104405                  CNTRLC: TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
4687  036622  104407                  CNTRLC: GTSWR                   ;ASK FOR NEW SWR VALUE
4688  036624  000137  033652          CNTRLC: JMP     $EOP+2          ;CONTINUE AT $EOP+2
4689  036630  000000          1$:     .WORD   0                       ;BUFFER FOR TEST NUMBER
4690  036632  020040    000   2$:     .ASCIZ  /  /                    ;TWO SPACES AND THE STOP MESSAGE
4691  036635    112   046525  044520  CMSG:   .ASCII  /JUMPING TO END-OF-PASS/<15><12>
4692  036642  043516  052040  020117
4693  036650  047105  026504  043117
4694  036656  050055  051501  006523
4695  036664     012
4696  036665    124   051505  047124          .ASCIZ  /TESTNO  PASSNO/<15><12>
4697  036672  020117  050040  051501
4698  036700  047123  006517  000012
4699                                          .EVEN
4700                                    .SBTTL  TYPE ROUTINE
4701
4702                                    ;;**********************************************************************
4703                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4704                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4705                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4706                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4707                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4708                                    ;*
4709                                    ;*CALL:
4710                                    ;*1) USING A TRAP INSTRUCTION
4711                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4712                                    ;*OR
4713                                    ;*      TYPE
4714                                    ;*      MESADR
4715                                    ;*
4716
4717  036706  105737  001157  $TYPE:   TSTB    $TPFLG          ;;IS THERE A TERMINAL?
4718  036712  100002                  BPL     1$              ;;BR IF YES
4719  036714  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
4720  036716  000430                  BR      3$              ;;LEAVE
4721  036720  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
4722  036722  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
4723  036726  122737  000001  001246  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4724  036734  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
4725  036736  132737  000100  001247  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
4726  036744  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
4727  036746  010037  036756          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
4728  036752  004737  037176          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
```

```
 4729  036756  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
 4730  036760  132737  000040  001247  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
 4731  036766  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
 4732  036770  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 4733  036772  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 4734  036774  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 4735  036776  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
 4736  037000  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
 4737  037004  000002                          RTI                     ;;RETURN
 4738  037006  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 4739  037012  001430                          BEQ     8$
 4740  037014  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 4741  037020  001006                          BNE     5$
 4742  037022  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
 4743  037024  104401                          TYPE                    ;;TYPE A CR AND LF
 4744  037026  001223                          $CRLF
 4745  037030  105037  037164                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
 4746  037034  000755                          BR      2$              ;;GET NEXT CHARACTER
 4747  037036  004737  037120          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 4748  037042  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 4749  037046  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
 4750  037050  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 4751                                                                  ;;AND THE NULL CHAR.
 4752  037054  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 4753  037060  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
 4754  037062  004737  037120                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
 4755  037066  105237  037164                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
 4756  037072  000770                          BR      7$              ;;LOOP
 4757
 4758                                  ;HORIZONTAL TAB PROCESSOR
 4759
 4760  037074  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
 4761  037100  004737  037120          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
 4762  037104  132737  000007  037164          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
 4763  037112  001372                          BNE     9$              ;;TAB STOP
 4764  037114  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
 4765  037116  000724                          BR      2$              ;;GET NEXT CHARACTER
 4766  037120  105777  142024          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 4767  037124  100375                          BPL     $TYPEC
 4768  037126  116677  000002  142016          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 4769  037134  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 4770  037142  001003                          BNE     1$              ;;BRANCH IF NO
 4771  037144  105037  037164                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 4772  037150  000406                          BR      $TYPEX          ;;EXIT
 4773  037152  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 4774  037160  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 4775  037162  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 4776  037164  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
 4777  037166  000207          $TYPEX: RTS     PC
 4778
 4779                                  .SBTTL  APT COMMUNICATIONS ROUTINE
 4780
 4781                                  ;;**************************************************************************
 4782  037170  112737  000001  037434  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 4783  037176  112737  000001  037432  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 4784  037204  000403                          BR      $ATYC
```

```
4785  037206  112737  000001  037434  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
4786  037214                          $ATYC:
4787  037214  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4788  037216  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4789  037220  105737  037432                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
4790  037224  001450                          BEQ     5$              ;;IF NOT:  BR
4791  037226  122737  000001  001246          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
4792  037234  001031                          BNE     3$              ;;IF NOT:  BR
4793  037236  132737  000100  001247          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4794  037244  001425                          BEQ     3$              ;;IF NOT:  BR
4795  037246  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
4796  037252  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4797  037260  005737  001226          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
4798  037264  001375                          BNE     1$              ;;IF NOT:  WAIT
4799  037266  010037  001242                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
4800  037272  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
4801  037274  001376                          BNE     2$
4802  037276  163700  001242                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
4803  037302  006200                          ASR     R0              ;;GET MESSAGF LNGTH IN WORDS
4804  037304  010037  001244                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
4805  037310  012737  000004  001226          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
4806  037316  000413                          BR      5$
4807  037320  017637  000004  037344  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
4808  037326  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
4809  037334  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
4810  037340  004737  036706                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
4811  037344  000000                  4$:     .WORD   0
4812  037346                          5$:
4813  037346  105737  037434          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
4814  037352  001416                          BEQ     12$             ;;IF NOT:  BR
4815  037354  005737  001246                  TST     $ENV            ;;RUNNING UNDER APT?
4816  037360  001413                          BEQ     12$             ;;IF NOT:  BR
4817  037362  005737  001226          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
4818  037366  001375                          BNE     11$             ;;IF NOT:  WAIT
4819  037370  017637  000004  001230          MOV     @4(SP),$FATAL   ;;GET ERROR #
4820  037376  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4821  037404  005237  001226                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
4822  037410  105037  037434          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
4823  037414  105037  037433                  CLRB    $LFLG           ;;CLEAR LOG FLAG
4824  037420  105037  037432                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
4825  037424  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4826  037426  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4827  037430  000207                          RTS     PC              ;;RETURN
4828  037432     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
4829  037433     000          $LFLG:  .BYTE   0               ;;LOG FLAG
4830  037434     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
4831          037436                  .EVEN
4832          000200          APTSIZE=200
4833          000001          APTENV=001
4834          000100          APTSPOOL=100
4835          000040          APTCSUP=040
4836                          .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
4837
4838                          ;;**************************************************************
4839                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
4840                          ;*BINARY-ASCII NUMBER AND TYPE IT.
```

```
                                        ;*CALL:
                                        ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
                                        ;*      TYPBN                   ;;TYPE IT
                                        ;*
037436  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
037440  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
037444  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
037446  112737  000060  037510  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
037454  006101                          ROL     R1              ;;GET THIS BIT
037456  001406                          BEQ     2$              ;;DONE?
037460  105537  037510                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
037464  104401  037510                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
037470  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
037472  000765                          BR      1$              ;;GO DO THE NEXT BIT
037474  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
037476  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
037504  012616                          MOV     (SP)+,(SP)
037506  000002                          RTI                     ;;RETURN TO USER
037510    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

037512  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
037516  116637  000001  037735          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
037524  112637  037737                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
037530  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
037534  000406                          BR      $TYPON
037536  112737  000001  037735  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
037544  112737  000006  037737          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
037552  112737  000005  037734  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
037560  010346                          MOV     R3,-(SP)        ;;SAVE R3
037562  010446                          MOV     R4,-(SP)        ;;SAVE R4
037564  010546                          MOV     R5,-(SP)        ;;SAVE R5
037566  113704  037737                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
```

```
4897  037572  005404                          NEG     R4
4898  037574  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
4899  037600  110437  037736                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
4900  037604  113704  037735                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
4901  037610  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
4902  037614  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
4903  037616  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
4904  037620  000404                          BR      3$              ;;GO DO MSB
4905  037622  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
4906  037624  006105                          ROL     R5
4907  037626  006105                          ROL     R5
4908  037630  010503                          MOV     R5,R3
4909  037632  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
4910  037634  105337  037736                  DECB    $OMODE          ;;TYPE THIS DIGIT?
4911  037640  100016                          BPL     7$              ;;BR IF NO
4912  037642  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
4913  037646  001002                          BNE     4$              ;;TEST FOR 0
4914  037650  005704                          TST     R4              ;;SUPPRESS THIS 0?
4915  037652  001403                          BEQ     5$              ;;BR IF YES
4916  037654  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
4917  037656  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
4918  037662  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
4919  037666  110337  037732                  MOVB    R3,8$           ;;SAVE FOR TYPING
4920  037672  104401  037732                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
4921  037676  105337  037734          7$:     DECB    $OCNT           ;;COUNT BY 1
4922  037702  003347                          BGT     2$              ;;BR IF MORE TO DO
4923  037704  002402                          BLT     6$              ;;BR IF DONE
4924  037706  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
4925  037710  000744                          BR      2$              ;;GO DO THE LAST DIGIT
4926  037712  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
4927  037714  012604                          MOV     (SP)+,R4        ;;RESTORE R4
4928  037716  012603                          MOV     (SP)+,R3        ;;RESTORE R3
4929  037720  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
4930  037726  012616                          MOV     (SP)+,(SP)
4931  037730  000002                          RTI                     ;;RETURN
4932  037732     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
4933  037733     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
4934  037734     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
4935  037735     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
4936  037736  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
4937                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4938
4939                                  ;;**************************************************************************
4940                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4941                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4942                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
4943                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4944                                  ;*REPLACED WITH SPACES.
4945                                  ;*CALL:
4946                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
4947                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
4948
4949  037740                          $TYPDS:
4950  037740  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4951  037742  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4952  037744  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

```
4953  037746  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4954  037750  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
4955  037752  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
4956  037756  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
4957  037762  100004                        BPL     1$              ;;BR IF INPUT IS POS.
4958  037764  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
4959  037766  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
4960  037774  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
4961  037776  012703  040154                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
4962  040002  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
4963  040006  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
4964  040010  016001  040144                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
4965  040014  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
4966  040016  002402                        BLT     4$              ;;BR IF DONE
4967  040020  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
4968  040022  000774                        BR      3$
4969  040024  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
4970  040026  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
4971  040030  001002                        BNE     5$              ;;FALL THROUGH IF 0
4972  040032  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
4973  040034  100407                        BMI     7$              ;;BR IF YES
4974  040036  106316                5$:     ASLB    (SP)            ;;MSD?
4975  040040  103003                        BCC     6$              ;;BR IF NO
4976  040042  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
4977  040050  052702  000060                6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
4978  040054  052702  000040                7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4979  040060  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4980  040062  005720                        TST     (R0)+           ;;JUST INCREMENTING
4981  040064  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
4982  040070  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
4983  040072  003002                        BGT     8$              ;;GO TO EXIT
4984  040074  010502                        MOV     R5,R2           ;;GET THE LSD
4985  040076  000764                        BR      6$              ;;GO CHANGE TO ASCII
4986  040100  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
4987  040102  100003                        BPL     9$              ;;BR IF NO
4988  040104  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
4989  040112  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
4990  040114  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
4991  040116  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
4992  040120  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
4993  040122  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4994  040124  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4995  040126  104401  040154                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
4996  040132  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
4997  040140  012616                        MOV     (SP)+,(SP)
4998  040142  000002                        RTI                     ;;RETURN TO USER
4999  040144  023420                $DTBL:  10000.
5000  040146  001750                        1000.
5001  040150  000144                        100.
5002  040152  000012                        10.
5003  040154  000004                $DBLK:  .BLKW   4
5004                                  .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5005
5006                                  ;;**************************************************************
5007                                  ;*SAVE R0-R5
5008                                  ;*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

040164                                  $SAVREG:
040164  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
040166  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
040170  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
040172  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
040174  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
040176  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
040200  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
040204  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
040210  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
040214  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
040220  000002                                  RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
040222                                  $RESREG:
040222  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
040226  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
040232  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
040236  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
040242  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
040244  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
040246  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
040250  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
040252  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
040254  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
040256  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

040260  104413                          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
040262  016601  000002                          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
040266  012705  040377                          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
040272  012704  000014                          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
040276  012703  177770                          MOV     #^C7,R3         ;;MASK
```

```
5065  040302  012100                    MOV     (R1)+,R0        ;;LOWER WORD
5066  040304  012101                    MOV     (R1)+,R1        ;;HIGH WORD
5067  040306  005002                    CLR     R2              ;;TERMINATOR
5068  040310  110245              1$:   MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
5069  040312  010002                    MOV     R0,R2           ;;GET THIS DIGIT
5070  040314  005304                    DEC     R4              ;;COUNT THIS CHARACTER
5071  040316  003007                    BGT     3$              ;;BR IF NOT THE LAST DIGIT
5072  040320  001405                    BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
5073  040322  005205                    INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5074  040324  010566  000002            MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
5075  040330  104414                    RESREG                  ;;RESTORE ALL REGISTERS
5076  040332  000207                    RTS     PC              ;;RETURN TO USER
5077  040334  006203              2$:   ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
5078  040336  006001              3$:   ROR     R1              ;;POSITION THE BINARY NUMBER FOR
5079  040340  006000                    ROR     R0              ;;      THE NEXT OCTAL DIGIT
5080  040342  006001                    ROR     R1
5081  040344  006000                    ROR     R0
5082  040346  006001                    ROR     R1
5083  040350  006000                    ROR     R0
5084  040352  040302                    BIC     R3,R2           ;;MASK OUT ALL JUNK
5085  040354  062702  000060            ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
5086  040360  000753                    BR      1$              ;;GO PUT IT IN THE DATA TABLE
5087  040362  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.

040400  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
040402  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
040406  005740                          TST     -(R0)           ;;BACKUP BY 2
040410  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
040412  006300                          ASL     R0              ;;POSITION FOR INDEXING
040414  016000  040434                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
040420  000200                          RTS     R0              ;;GO TO ROUTINE

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

040422  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
040424  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
040432  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
040434  040422                  $TRPAD: .WORD   $TRAP2
040436  036706                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
040440  037536                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
040442  037512                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
040444  037552                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
040446  037740                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
040450  037436                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

040452  035634                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

040454  035564                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
040456  036106                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
040460  036226                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
040462  040164                          $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
040464  040222                          $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;**********************************************************************
                                        ;POWER DOWN ROUTINE
040466  012737  040644  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
040474  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
040502  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
040504  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
040506  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
040510  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
040512  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
```

```
5144  040514  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
5145  040516  017746  140416                  MOV     @$WR,-(SP)        ;;PUSH @$WR ON STACK
5146  040522  010637  040650                  MOV     SP,$SAVR6         ;;SAVE SP
5147  040526  012737  040540  000024          MOV     #$PWRUP,@#PWRVEC  ;;SET UP VECTOR
5148  040534  000000                          HALT
5149  040536  000776                          BR      .-2               ;;HANG UP
5150
5151                                  ;;*****************************************************************
5152                                  ;POWER UP ROUTINE
5153  040540  012737  040644  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
5154  040546  013706  040650                  MOV     $SAVR6,SP         ;;GET SP
5155  040552  005037  040650                  CLR     $SAVR6            ;;WAIT LOOP FOR THE TTY
5156  040556  005237  040650          1$:     INC     $SAVR6            ;;WAIT FOR THE INC
5157  040562  001375                          BNE     1$                ;;OF  WORD
5158  040564  012677  140350                  MOV     (SP)+,@$WR        ;;POP STACK INTO @$WR
5159  040570  012605                          MOV     (SP)+,R5          ;;POP STACK INTO R5
5160  040572  012604                          MOV     (SP)+,R4          ;;POP STACK INTO R4
5161  040574  012603                          MOV     (SP)+,R3          ;;POP STACK INTO R3
5162  040576  012602                          MOV     (SP)+,R2          ;;POP STACK INTO R2
5163  040600  012601                          MOV     (SP)+,R1          ;;POP STACK INTO R1
5164  040602  012600                          MOV     (SP)+,R0          ;;POP STACK INTO R0
5165  040604  012737  040466  000024          MOV     #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
5166  040612  012737  000340  000026          MOV     #340,@#PWRVEC+2   ;;PRIO:7
5167  040620  104401                          TYPE                      ;;REPORT THE POWER FAILURE
5168  040622  040652                  $PWRMG: .WORD   PWRMSG             ;;POWER FAIL MESSAGE POINTER
5169  040624  012716                          MOV     (PC)+,(SP)        ;;RESTART AT RESTRT
5170  040626  020462                  $PWRAD: .WORD   RESTRT             ;;RESTART ADDRESS
5171  040630  042766  000020  000002          BIC     #20,2(SP)         ;;CLEAR "T" BIT
5172  040636  005037  034130                  CLR     $TBIT             ;;CLEAR THE "T" BIT FLAG
5173  040642  000002                          RTI
5174  040644  000000                  $ILLUP: HALT                      ;;THE POWER UP SEQUENCE WAS STARTED
5175  040646  000776                          BR      .-2               ;; BEFORE THE POWER DOWN WAS COMPLETE
5176  040650  000000                  $SAVR6: 0                         ;;PUT THE SP HERE
5177  040652  006412  050040  053517  PWRMSG: .ASCIZ  <12><15>? POWER FAILURE - RESTARTING ?<12><15>
5178  040660  051105  043040  044501
5179  040666  052514  042522  026440
5180  040674  051040  051505  040524
5181  040702  052122  047111  020107
5182  040710  006412     000
5183          040714                          .EVEN
5184
5185
```

```
                                        .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
040714 047125 054105 042520 EM1:    .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
040722 052103 042105 041440
040730 052520 052040 040522
040736 020120 047524 046040
040744 041517 020056 030060
040752 000064
040754 047125 054105 042520 EM2:    .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
040762 052103 042105 046440
040770 046505 020056 043515
040776 052115 020056 051124
041004 050101 052040 020117
041012 047514 027103 031040
041020 030065    000
041023    120 044522 051117 EM3:    .ASCIZ  /PRIORITY BITS SET WRONG IN PSW/
041030 052111 020131 044502
041036 051524 051440 052105
041044 053440 047522 043516
041052 044440 020116 051520
041060 000127
041062 047515 042504 041040 EM4:    .ASCIZ  /MODE BITS SET WRONG IN PSW/
041070 052111 020123 042523
041076 020124 051127 047117
041104 020107 047111 050040
041112 053523    000
041115    104 040525 020114 EM5:    .ASCIZ  /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
041122 042101 051104 051505
041130 044523 043516 041040
041136 052105 042527 047105
041144 044040 023111 047514
041152 041040 052131 051505
041160 047440 020106 051520
041166 000127
041170 042513 047122 046105 EM6:    .ASCIZ  /KERNEL R6 CHANGED BY WRITING USER R6/
041176 051040 020066 044103
041204 047101 042507 020104
041212 054502 053440 044522
041220 044524 043516 052440
041226 042523 020122 033122
041234    000
041235    101 046440 046505 EM7:    .ASCIZ  /A MEMORY MGMT. REG. TIMED OUT/
041242 051117 020131 043515
041250 052115 020056 042522
041256 027107 052040 046511
041264 042105 047440 052125
041272    000
041273    123 046525 040515 EM10:   .ASCIZ  /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
041300 054522 047440 020106
041306 042515 027115 046440
041314 046507 027124 051040
041322 043505 020056 044524
041330 042515 052517 051524
041336    000
041337    115 046505 020056 EM11:   .ASCIZ  /MEM. MGMT. REG. WOULD NOT CLEAR/
041344 043515 052115 020056
041352 042522 027107 053440
```

```
 5242  041360  052517  042114  047040
 5243  041366  052117  041440  042514
 5244  041374  051101     000
 5245  041377     115  046505  020056  EM12:   .ASCIZ  /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
 5246  041404  043515  052115  020056
 5247  041412  042522  027107  041040
 5248  041420  052111  020123  047516
 5249  041426  020124  042523  020124
 5250  041434  047503  051122  041505
 5251  041442  046124  000131
 5252  041446  051123  020060  043105  EM13:   .ASCIZ  /SR0 EFFECTED BY WRITE TO PSW/
 5253  041454  042506  052103  042105
 5254  041462  041040  020131  051127
 5255  041470  052111  020105  047524
 5256  041476  050040  053523     000
 5257  041503     123  030522  042040  EM14:   .ASCIZ  /SR1 DID NOT READ ALL ZEROS/
 5258  041510  042111  047040  052117
 5259  041516  051040  040505  020104
 5260  041524  046101  020114  042532
 5261  041532  047522  000123
 5262  041536  052504  046101  040440  EM15:   .ASCIZ  /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
 5263  041544  042104  042522  051523
 5264  041552  047111  020107  042502
 5265  041560  053524  042505  020116
 5266  041566  054502  042524  020123
 5267  041574  043117  050040  051101
 5268  041602  047440  020122  042120
 5269  041610  000122
 5270  041612  052504  046101  040440  EM16:   .ASCIZ  /DUAL ADDRESSING BETWEEN PAR-PDR'S/
 5271  041620  042104  042522  051523
 5272  041626  047111  020107  042502
 5273  041634  053524  042505  020116
 5274  041642  040520  026522  042120
 5275  041650  023522  000123
 5276  041654  044120  051531  041511  EM17:   .ASCIZ  /PHYSICAL ADDRESS FORMED WRONG/
 5277  041662  046101  040440  042104
 5278  041670  042522  051523  043040
 5279  041676  051117  042515  020104
 5280  041704  051127  047117  000107
 5281  041712  044120  051531  020056  EM20:   .ASCIZ  /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
 5282  041720  042101  051104  020056
 5283  041726  047506  046522  042105
 5284  041734  053440  047522  043516
 5285  041742  044440  020116  042522
 5286  041750  047514  040503  042524
 5287  041756  046440  042117  000105
 5288  041764  026527  044502  020124  EM21:   .ASCIZ  /W-BIT DID NOT GET SET IN PDR/
 5289  041772  044504  020104  047516
 5290  042000  020124  042507  020124
 5291  042006  042523  020124  047111
 5292  042014  050040  051104     000
 5293  042021     127  041055  052111  EM22:   .ASCIZ  /W-BIT SET IN MORE THAN ONE PDR/
 5294  042026  051440  052105  044440
 5295  042034  020116  047515  042522
 5296  042042  052040  040510  020116
 5297  042050  047117  020105  042120
```

```
 5298  042056  000122
 5299  042060  026527  044502  020124  EM23:   .ASCIZ  /W-BIT NOT CLEARED BY WRITING TO PDR/
 5300  042066  047516  020124  046103
 5301  042074  040505  042522  020104
 5302  042102  054502  053440  044522
 5303  042110  044524  043516  052040
 5304  042116  020117  042120  000122
 5305  042124  051127  052111  047111  EM24:   .ASCIZ  /WRITING SR0 SET W-BIT IN K1PDR7/
 5306  042132  020107  051123  020060
 5307  042140  042523  020124  026527
 5308  042146  044502  020124  047111
 5309  042154  045440  050111  051104
 5310  042162  000067
 5311  042164  026527  044502  020124  EM25:   .ASCIZ  /W-BIT GOT SET DURING TIMEOUT ABORT/
 5312  042172  047507  020124  042523
 5313  042200  020124  052504  044522
 5314  042206  043516  052040  046511
 5315  042214  047505  052125  040440
 5316  042222  047502  052122     000
 5317  042227     115  046505  051117  EM26:   .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
 5318  042234  020131  043515  052115
 5319  042242  020056  041501  042503
 5320  042250  051523  040440  047502
 5321  042256  052122  042040  042111
 5322  042264  047040  052117  047440
 5323  042272  041503  051125     000
 5324  042277     101  041503  051505  EM27:   .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
 5325  042304  020123  051105  047522
 5326  042312  020122  044504  020104
 5327  042320  047516  020124  041101
 5328  042326  051117  020124  047111
 5329  042334  052123  052522  052103
 5330  042342  047511  000116
 5331  042346  051123  020060  044504  EM30:   .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
 5332  042354  020104  047516  020124
 5333  042362  042522  047520  052122
 5334  042370  040440  041503  051505
 5335  042376  020123  051105  047522
 5336  042404  020122  047503  051122
 5337  042412  041505  046124  000131
 5338  042420  044504  020104  047516  EM31:   .ASCIZ  /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
 5339  042426  020124  047514  045503
 5340  042434  050125  041440  051117
 5341  042442  042522  052103  053040
 5342  042450  051111  052524  046101
 5343  042456  040440  042104  027122
 5344  042464     000
 5345  042465     120  043501  020105  EM32:   .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
 5346  042472  043514  044124  020056
 5347  042500  041101  051117  020124
 5348  042506  041517  052503  051122
 5349  042514  042105  053440  042510
 5350  042522  020116  052111  051440
 5351  042530  047510  046125  047104
 5352  042536  052047  044040  053101
 5353  042544  000105
```

```
 5354  042546  040520  042507  046040  EM33:   .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
 5355  042554  052107  027110  040440
 5356  042562  047502  052122  042040
 5357  042570  042111  047040  052117
 5358  042576  047440  041503  051125
 5359  042604  053440  042510  020116
 5360  042612  052111  051440  047510
 5361  042620  046125  020104  040510
 5362  042626  042526     000
 5363  042631     123  030122  042040  EM34:   .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
 5364  042636  042111  047040  052117
 5365  042644  051040  050105  051117
 5366  042652  020124  040520  042507
 5367  042660  046040  052107  027110
 5368  042666  040440  047502  052122
 5369  042674  041440  051117  042522
 5370  042702  052103  054514     000
 5371  042707     123  030122  047440  EM37:   .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
 5372  042714  020122  051123  020062
 5373  042722  044103  047101  042507
 5374  042730  020104  054502  040440
 5375  042736  051440  041505  047117
 5376  042744  020104  041101  051117
 5377  042752  000124
 5378  042754  051123  020060  051117  EM40:   .ASCIZ  /SR0 OR SR2 WERE NOT "RESET" BY A RESET/
 5379  042762  051440  031122  053440
 5380  042770  051105  020105  047516
 5381  042776  020124  051042  051505
 5382  043004  052105  020042  054502
 5383  043012  040440  051040  051505
 5384  043020  052105     000
 5385  043023     123  031122  047040  EM41:   .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
 5386  043030  052117  052040  040522
 5387  043036  045503  047111  020107
 5388  043044  047503  051122  041505
 5389  043052  046124  000131
 5390  043056  044504  020104  047516  EM42:   .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
 5391  043064  020124  051124  050101
 5392  043072  052040  051110  020125
 5393  043100  042513  047122  046105
 5394  043106  051440  040520  042503
 5395  043114     000
 5396  043115     113  020124  051105  EM43:   .ASCIZ  /KT ERROR NOT SERVICED ON TIMEOUT ERROR/
 5397  043122  047522  020122  047516
 5398  043130  020124  042523  053122
 5399  043136  041511  042105  047440
 5400  043144  020116  044524  042515
 5401  043152  052517  020124  051105
 5402  043160  047522  000122
 5403  043164  051123  020060  051117  EM44:   .ASCIZ  /SR0 OR SR2 CHANGED BY TIMEOUT ERROR/
 5404  043172  051440  031122  041440
 5405  043200  040510  043516  042105
 5406  043206  041040  020131  044524
 5407  043214  042515  052517  020124
 5408  043222  051105  047522  000122
 5409  043230  051105  047522  020122  EM45:   .ASCIZ  /ERROR DURING "DOUBLE ERROR" (KT & TIMEOUT)/
```

```
 5410  043236  052504  044522  043516
 5411  043244  021040  047504  041125
 5412  043252  042514  042440  051122
 5413  043260  051117  020042  045450
 5414  043266  020124  020046  044524
 5415  043274  042515  052517  024524
 5416  043302     000
 5417  043303     115  050106  020111  EM46:   .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
 5418  043310  047111  052123  052522
 5419  043316  052103  047511  020116
 5420  043324  052520  044123  042105
 5421  043332  053440  047522  043516
 5422  043340  042040  052101  000101
 5423  043346  052115  044520  044440  EM47:   .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
 5424  043354  051516  051124  041525
 5425  043362  044524  047117  046040
 5426  043370  040517  042504  020104
 5427  043376  051127  047117  020107
 5428  043404  040504  040524     000
 5429  043411     123  040524  045503  EM50:   .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
 5430  043416  047040  052117  050040
 5431  043424  051525  042510  020104
 5432  043432  054502  046440  050106
 5433  043440  026511  052115  044520
 5434  043446     000
 5435  043447     113  051105  042516  EM51:   .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
 5436  043454  020114  040520  042507
 5437  043462  040440  041503  051505
 5438  043470  020123  047111  052123
 5439  043476  040505  020104  043117
 5440  043504  052440  042523  035122
 5441  043512  046440  050106  026511
 5442  043520  052115  044520     000
 5443  043525     127  047522  043516  EM52:   .ASCIZ  /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
 5444  043532  050040  051104  051447
 5445  043540  051040  043105  051105
 5446  043546  047105  042503  020104
 5447  043554  044127  046111  020105
 5448  043562  047111  051040  046105
 5449  043570  041517  052101  020105
 5450  043576  047515  042504     000
 5451  043603     115  050106  020104  EM53:   .ASCIZ  /MFPD INSTRUCTION PUSHED WRONG DATA/
 5452  043610  047111  052123  052522
 5453  043616  052103  047511  020116
 5454  043624  052520  044123  042105
 5455  043632  053440  047522  043516
 5456  043640  042040  052101  000101
 5457  043646  052123  041501  020113  EM54:   .ASCIZ  /STACK NOT PUSHED BY MFPD-MTPD/
 5458  043654  047516  020124  052520
 5459  043662  044123  042105  041040
 5460  043670  020131  043115  042120
 5461  043676  046455  050124  000104
 5462  043704  040520  020122  051117  EM55:   .ASCIZ  /PAR OR PDR CHANGED BY A RESET/
 5463  043712  050040  051104  041440
 5464  043720  040510  043516  042105
 5465  043726  041040  020131  020101
```

```
 5466  043734  042522  042523  000124
 5467  043742  051520  020127  044103  EM56:   .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
 5468  043750  047101  042507  020104
 5469  043756  054502  040440  020116
 5470  043764  052122  020111  047111
 5471  043772  052440  042523  020122
 5472  044000  047515  042504     000
 5473
 5474  044005     117  042114  050040  DH1:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  TESTNO  ERRORPC/
 5475  044012  020103  047440  042114
 5476  044020  050040  053523  051040
 5477  044026  020066  040527  020123
 5478  044034  052040  051505  047124
 5479  044042  020117  042440  051122
 5480  044050  051117  041520     000
 5481  044055     117  042114  050040  DH2:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC/
 5482  044062  020103  047440  042114
 5483  044070  050040  053523  051040
 5484  044076  020066  040527  020123
 5485  044104  051440  030122  020040
 5486  044112  020040  051440  031122
 5487  044120  020040  020040  052040
 5488  044126  051505  047124  020117
 5489  044134  042440  051122  051117
 5490  044142  041520     000
 5491  044145     127  047522  042524  DH3:    .ASCIZ  /WROTE   READ    TESTNO  ERRORPC/
 5492  044152  020040  051040  040505
 5493  044160  020104  020040  052040
 5494  044166  051505  047124  020117
 5495  044174  042440  051122  051117
 5496  044202  041520     000
 5497  044205     101  042104  042522  DH7:    .ASCIZ  /ADDRESS TESTNO  ERRORPC/
 5498  044212  051523  052040  051505
 5499  044220  047124  020117  042440
 5500  044226  051122  051117  041520
 5501  044234     000
 5502  044235     122  043505  051511  DH10:   .ASCII  /REGISTER-ADDRS  NUM  OF/<CRLF>
 5503  044242  042524  026522  042101
 5504  044250  051104  020123  047040
 5505  044256  046525  020040  043117
 5506  044264     200
 5507  044265     101  042116  042455          .ASCIZ  /AND-ED  OR-ED   TIMOUTS TESTNO  ERRORPC/
 5508  044272  020104  047440  026522
 5509  044300  042105  020040  052040
 5510  044306  046511  052517  051524
 5511  044314  052040  051505  047124
 5512  044322  020117  042440  051122
 5513  044330  051117  041520     000
 5514  044335     122  043505  051511  DH11:   .ASCII  /REGISTR READ    READ-(BINARY)/<CRLF>
 5515  044342  051124  051040  040505
 5516  044350  020104  020040  051040
 5517  044356  040505  026504  041050
 5518  044364  047111  051101  024531
 5519  044372     200
 5520  044373     101  042104  042522          .ASCIZ  /ADDRESS (OCTAL) 5432109876543210  TESTNO  ERRORPC/
 5521  044400  051523  024040  041517
```

```
5522  044406  040524  024514  032440
5523  044414  031464  030462  034460
5524  044422  033470  032466  031464
5525  044430  030462  020060  052040
5526  044436  051505  047124  020117
5527  044444  042440  051122  051117
5528  044452  041520     000
5529  044455     122  043505  051511  DH12:   .ASCII  /REGISTR WROTE   READ    READ-(BINARY)/<CRLF>
5530  044462  051124  053440  047522
5531  044470  042524  020040  051040
5532  044476  040505  020104  020040
5533  044504  051040  040505  026504
5534  044512  041050  047111  051101
5535  044520  024531     200
5536  044523     101  042104  042522          .ASCIZ  /ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC/
5537  044530  051523  024040  041517
5538  044536  040524  024514  024040
5539  044544  041517  040524  024514
5540  044552  032440  031464  030462
5541  044560  034460  033470  032466
5542  044566  031464  030462  020060
5543  044574  052040  051505  047124
5544  044602  020117  042440  051122
5545  044610  051117  041520     000
5546  044615     122  040505  020104  DH13:   .ASCIZ  /READ    TESTNO  ERRORPC/
5547  044622  020040  052040  051505
5548  044630  047124  020117  042440
5549  044636  051122  051117  041520
5550  044644     000
5551  044645     120  051101  050055  DH16:   .ASCII  /PAR-PDR PAR-PDR/<CRLF>
5552  044652  051104  050040  051101
5553  044660  050055  051104     200
5554  044665     103  042514  051101          .ASCIZ  /CLEARED EFFECTD EXPECTD RECEIVD TESTNO  ERRORPC/
5555  044672  042105  042440  043106
5556  044700  041505  042124  042440
5557  044706  050130  041505  042124
5558  044714  051040  041505  044505
5559  044722  042126  052040  051505
5560  044730  047124  020117  042440
5561  044736  051122  051117  041520
5562  044744     000
5563  044745     120  054510  044523  DH17:   .ASCII  /PHYSICL VIRTUAL/<CRLF>
5564  044752  046103  053040  051111
5565  044760  052524  046101     200
5566  044765     101  042104  042522          .ASCIZ  /ADDRESS ADDRESS KIPAR4  TESTNO  ERRORPC/
5567  044772  051523  040440  042104
5568  045000  042522  051523  045440
5569  045006  050111  051101  020064
5570  045014  052040  051505  047124
5571  045022  020117  042440  051122
5572  045030  051117  041520     000
5573  045035     120  054510  044523  DH20:   .ASCII  /PHYSICL PAR 4   PAR 5/<CRLF>
5574  045042  046103  050040  051101
5575  045050  032040  020040  050040
5576  045056  051101  032440     200
5577  045063     101  042104  042522          .ASCIZ  /ADDRESS VBA     VBA     PAR 4   PAR 5   PSW     TESTNO  ERRORPC/
```

```
 5578  045070  051523  053040  040502
 5579  045076  020040  020040  053040
 5580  045104  040502  020040  020040
 5581  045112  050040  051101  032040
 5582  045120  020040  050040  051101
 5583  045126  032440  020040  050040
 5584  045134  053523  020040  020040
 5585  045142  052040  051505  047124
 5586  045150  020117  042440  051122
 5587  045156  051117  041520     000
 5588  045163     120  051104  020040  DH21:   .ASCII  /PDR     VIRTUAL/<CRLF>
 5589  045170  020040  053040  051111
 5590  045176  052524  046101     200
 5591  045203     124  051505  042524          .ASCIZ  /TESTED  ADDRESS TESTNO  ERRORPC/
 5592  045210  020104  040440  042104
 5593  045216  042522  051523  052040
 5594  045224  051505  047124  020117
 5595  045232  042440  051122  051117
 5596  045240  041520     000
 5597  045243     120  051104  044440  DH22:   .ASCII  /PDR IN  PDR     VIRTUAL/
 5598  045250  020116  050040  051104
 5599  045256  020040  020040  053040
 5600  045264  051111  052524  046101
 5601  045272  051105  047522  020122          .ASCIZ  /ERROR   TESTED  ADDRESS TESTNO  ERRORPC/
 5602  045300  020040  042524  052123
 5603  045306  042105  020040  042101
 5604  045314  051104  051505  020123
 5605  045322  042524  052123  047516
 5606  045330  020040  051105  047522
 5607  045336  050122  000103
 5608  045342  042120  020122  020040  DH23:   .ASCIZ  /PDR     TESTNO  ERRORPC/
 5609  045350  020040  042524  052123
 5610  045356  047516  020040  051105
 5611  045364  047522  050122  000103
 5612  045372  042120  020122  040527  DH24:   .ASCIZ  /PDR WAS EXPECTD TESTNO  ERRORPC/
 5613  045400  020123  054105  042520
 5614  045406  052103  020104  042524
 5615  045414  052123  047516  020040
 5616  045422  051105  047522  050122
 5617  045430  000103
 5618  045432  042120  020122  020064  DH26:   .ASCIZ  /PDR 4   PSW     TESTNO  ERRORPC/
 5619  045440  020040  051520  020127
 5620  045446  020040  020040  042524
 5621  045454  052123  047516  020040
 5622  045462  051105  047522  050122
 5623  045470  000103
 5624  045472  051123  020060  040527  DH30:   .ASCIZ  /SRO WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
 5625  045500  020123  054105  042520
 5626  045506  052103  020104  042120
 5627  045514  020122  020064  020040
 5628  045522  051520  020127  020040
 5629  045530  020040  042524  052123
 5630  045536  047516  020040  051105
 5631  045544  047522  050122  000103
 5632  045552  051123  020062  040527  DH31:   .ASCIZ  /SR2 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
 5633  045560  020123  054105  042520
```

```
 5634  045566  052103  020104  042120
 5635  045574  020122  020064  020040
 5636  045602  051520  020127  020040
 5637  045610  020040  042524  052123
 5638  045616  047516  020040  051105
 5639  045624  047522  050122  000103
 5640  045632  027126  027102  027101  DH32:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC/
 5641  045640  020040  044513  042120
 5642  045646  032122  020040  051123
 5643  045654  020060  040527  020123
 5644  045662  051123  020062  040527
 5645  045670  020123  042524  052123
 5646  045676  047516  020040  051105
 5647  045704  047522  050122  000103
 5648  045712  027126  027102  027101  DH33:   .ASCIZ  /V.B.A.  KIPDR4  TESTNO  ERRORPC/
 5649  045720  020040  044513  042120
 5650  045726  032122  020040  042524
 5651  045734  052123  047516  020040
 5652  045742  051105  047522  050122
 5653  045750  000103
 5654  045752  027126  027102  027101  DH34:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC/
 5655  045760  020040  044513  042120
 5656  045766  032122  020040  051123
 5657  045774  020060  040527  020123
 5658  046002  054105  042520  052103
 5659  046010  020104  042524  052123
 5660  046016  047516  020040  051105
 5661  046024  047522  050122  000103
 5662  046032  027126  027102  027101  DH35:   .ASCIZ  /V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC/
 5663  046040  020040  044513  042120
 5664  046046  032122  020040  051123
 5665  046054  020062  040527  020123
 5666  046062  054105  042520  052103
 5667  046070  020104  042524  052123
 5668  046076  047516  020040  051105
 5669  046104  047522  050122  000103
 5670  046112  051123  020062  040527  DH36:   .ASCIZ  /SR2 WAS EXPECTD TESTNO  ERRORPC/
 5671  046120  020123  054105  042520
 5672  046126  052103  020104  042524
 5673  046134  052123  047516  020040
 5674  046142  051105  047522  050122
 5675  046150  000103
 5676  046152  044506  051522  020124  DH37:   .ASCII  /FIRST ABORT     SECOND ABORT/<CRLF>
 5677  046160  041101  051117  020124
 5678  046166  020040  020040  042523
 5679  046174  047503  042116  040440
 5680  046202  047502  052122     200
 5681  046207     123  030122  053440          .ASCIZ  /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC/
 5682  046214  051501  051440  031122
 5683  046222  053440  051501  051440
 5684  046230  030122  053440  051501
 5685  046236  051440  031122  053440
 5686  046244  051501  052040  051505
 5687  046252  047124  020117  042440
 5688  046260  051122  051117  041520
 5689  046266     000
```

```
 5690  046267     123  030122  053440  DH40:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
 5691  046274  051501  051440  031122
 5692  046302  053440  051501  052040
 5693  046310  051505  047124  020117
 5694  046316  042440  051122  051117
 5695  046324  041520     000
 5696  046327     120  053523  053440  DH42:   .ASCIZ  /PSW WAS R6 WAS  TESTNO  ERRORPC/
 5697  046334  051501  051040  020066
 5698  046342  040527  020123  052040
 5699  046350  051505  047124  020117
 5700  046356  042440  051122  051117
 5701  046364  041520     000
 5702  046367     105  050130  041505  DH44:   .ASCII  /EXPECTED         RECEIVED/<CRLF>
 5703  046374  042524  020104  020040
 5704  046402  020040  020040  020040
 5705  046410  042522  042503  053111
 5706  046416  042105     200
 5707  046421     123  030122  020040          .ASCIZ  /SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC/
 5708  046426  020040  051440  031122
 5709  046434  020040  020040  051440
 5710  046442  030122  053440  051501
 5711  046450  051440  031122  053440
 5712  046456  051501  052040  051505
 5713  046464  047124  020117  042440
 5714  046472  051122  051117  041520
 5715  046500     000
 5716  046501     105  050130  041505  DH45:   .ASCII  /EXPECTED:/<CRLF>
 5717  046506  042524  035104     200
 5718  046513     120  053523  020040          .ASCII  /PSW     PC      SR0     SR2/<CRLF>
 5719  046520  020040  050040  020103
 5720  046526  020040  020040  051440
 5721  046534  030122  020040  020040
 5722  046542  051440  031122     200
 5723  046547     061  030067  030460          .ASCII  /170017  (3$+4)  020147  (3$)/<CRLF>
 5724  046554  020067  024040  022063
 5725  046562  032053  020051  030040
 5726  046570  030062  032061  020067
 5727  046576  024040  022063  100051
 5728  046604  042522  042503  053111          .ASCII  /RECEIVED:/<CRLF>
 5729  046612  042105  100072
 5730  046616  051520  020127  020040          .ASCIZ  /PSW     PC      SR0     SR2     TESTNO  ERRORPC/
 5731  046624  020040  041520  020040
 5732  046632  020040  020040  051123
 5733  046640  020040  020040  020040
 5734  046646  051[illegible]  020062  020040
 5735  046654  020[illegible]  042524  052123
 5736  046662  047516  020040  051105
 5737  046670  047522  050122  000103
 5738  046676  040504  040524  020040  DH46:   .ASCII  /DATA    DATA/<CRLF>
 5739  046704  020040  040504  040524
 5740  046712     200
 5741  046713     105  050130  041505          .ASCIZ  /EXPECTD RECEIVD TESTNO  ERRORPC/
 5742  046720  042124  051040  041505
 5743  046726  044505  042126  052040
 5744  046734  051505  047124  020117
 5745  046742  042440  051122  051117
```

```
 5746  046750  041520     000
 5747  046753     124  051505  047124  DH50:   .ASCIZ  /TESTNO  ERRORPC/
 5748  046760  020117  042440  051122
 5749  046766  051117  041520     000
 5750  046773     123  030122  053440  DH51:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
 5751  047000  051501  051440  031122
 5752  047006  053440  051501  052040
 5753  047014  051505  047124  020117
 5754  047022  042440  051122  051117
 5755  047030  041520     000
 5756  047033     120  054510  044523  DH52:   .ASCII  /PHYSICL PAR 4/<CRLF>
 5757  047040  046103  050040  051101
 5758  047046  032040     200
 5759  047051     101  042104  042522          .ASCIZ  /ADDRESS V.B.A.  PAR 4   SR0 WAS SR2 WAS PSW     TESTNO  ERRORPC/
 5760  047056  051523  053040  041056
 5761  047064  040456  020056  050040
 5762  047072  051101  032040  020040
 5763  047100  051440  030122  053440
 5764  047106  051501  051440  031122
 5765  047114  053440  051501  050040
 5766  047122  053523  020040  020040
 5767  047130  052040  051505  047124
 5768  047136  020117  042440  051122
 5769  047144  051117  041520     000
 5770  047151     120  053523  053440  DH56:   .ASCIZ  /PSW WAS EXPECTD TESTNO  ERRORPC/
 5771  047156  051501  042440  050130
 5772  047164  041505  042124  052040
 5773  047172  051505  047124  020117
 5774  047200  042440  051122  051117
 5775  047206  041520     000
 5776
 5777          047212                          .EVEN
 5778
 5779  047212  001266  001270  001264  DT1:    .WORD   TRAPPC,TRAPPS,WASR6,TESTNO,$ERRPC,0
 5780  047220  001262  001116  000000
 5781  047226  001266  001270  001264  DT2:    .WORD   TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5782  047234  001272  001274  001262
 5783  047242  001116  000000
 5784  047246  001162  001164  001262  DT3:    .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
 5785  047254  001116  000000
 5786  047260  001162  001262  001116  DT7:    .WORD   $REG0,TESTNO,$ERRPC,0
 5787  047266  000000
 5788  047270  001300  001302  001304  DT10:   .WORD   ANDADR,GRADR,TONUM,TESTNO,$ERRPC,0
 5789  047276  001262  001116  000000
 5790  047304  001162  001164  001164  DT11:   .WORD   $REG0,$REG1,$REG1,TESTNO,$ERRPC,0
 5791  047312  001262  001116  000000
 5792  047320  001162  001164  001166  DT12:   .WORD   $REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
 5793  047326  001166  001262  001116
 5794  047334  000000
 5795  047336  001162  001262  001116  DT13:   .WORD   $REG0,TESTNO,$ERRPC,0
 5796  047344  000000
 5797  047346  001162  001164  001174  DT16:   .WORD   $REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
 5798  047354  001166  001262  001116
 5799  047362  000000
 5800  047364  001312  001306  001172  DT17:   .WORD   PBALO,VIRT1,$REG4,TESTNO,$ERRPC,0
 5801  047372  001262  001116  000000
```

```
 5802  047400  001312  001306  001310  DT20:   .WORD   PBALO,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
 5803  047406  001172  001174  001176
 5804  047414  001262  001116  000000
 5805  047422  001174  001170  001262  DT21:   .WORD   $REG5,$REG3,TESTNO,$ERRPC,0
 5806  047430  001116  000000
 5807  047434  001162  001174  001170  DT22:   .WORD   $REG0,$REG5,$REG3,TESTNO,$ERRPC,0
 5808  047442  001262  001116  000000
 5809  047450  001174  001262  001116  DT23:   .WORD   $REG5,TESTNO,$ERRPC,0
 5810  047456  000000
 5811  047460  001166  001164  001262  DT24:   .WORD   $REG2,$REG1,TESTNO,$ERRPC,0
 5812  047466  001116  000000
 5813  047472  001166  001176  001262  DT26:   .WORD   $REG2,$TMP0,TESTNO,$ERRPC,0
 5814  047500  001116  000000
 5815  047504  001272  001170  001166  DT30:   .WORD   WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
 5816  047512  001176  001262  001116
 5817  047520  000000
 5818  047522  001274  001172  001166  DT31:   .WORD   WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
 5819  047530  001176  001262  001116
 5820  047536  000000
 5821  047540  001162  001172  001272  DT32:   .WORD   $REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5822  047546  001274  001262  001116
 5823  047554  000000
 5824  047556  001162  001172  001262  DT33:   .WORD   $REG0,$REG4,TESTNO,$ERRPC,0
 5825  047564  001116  000000
 5826  047570  001162  001172  001272  DT34:   .WORD   $REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
 5827  047576  001166  001262  001116
 5828  047604  000000
 5829  047606  001162  001172  001274  DT35:   .WORD   $REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
 5830  047614  001170  001262  001116
 5831  047622  000000
 5832  047624  001274  001164  001262  DT36:   .WORD   WASSR2,$REG1,TESTNO,$ERRPC,0
 5833  047632  001116  000000
 5834  047636  001176  001202  001272  DT37:   .WORD   $TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5835  047644  001274  001262  001116
 5836  047652  000000
 5837  047654  001272  001274  001262  DT40:   .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
 5838  047662  001116  000000
 5839  047666  001164  001166  001262  DT42:   .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
 5840  047674  001116  000000
 5841  047700  001162  001164  001272  DT44:   .WORD   $REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5842  047706  001274  001262  001116
 5843  047714  000000
 5844  047716  001164  001170  001272  DT45:   .WORD   $REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
 5845  047724  001274  001262  001116
 5846  047732  000000
 5847  047734  001162  001164  001262  DT46:   .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
 5848  047742  001116  000000
 5849  047746  001262  001116  000000  DT50:   .WORD   TESTNO,$ERRPC,0
 5850  047754  001272  001274  001262  DT51:   .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
 5851  047762  001116  000000
 5852  047766  001312  001306  001172  DT52:   .WORD   PBALO,VIRT1,$REG4,WASSR0,WASSR2,$TMP0,TESTNO,$ERRPC,0
 5853  047774  001272  001274  001176
 5854  050002  001262  001116  000000
 5855  050010  001164  001166  001262  DT56:   .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
 5856  050016  001116  000000
 5857
```

```
 5858  050022      000      000      000  DF1:    .BYTE   0,0,0,0,0
 5859  050025      000      000
 5860  050027      000      000      000  DF2:    .BYTE   0,0,0,0,0,0,0
 5861  050032      000      000      000
 5862  050035      000
 5863  050036      000      000      000  DF3:    .BYTE   0,0,0,0
 5864  050041      000
 5865  050042      000      000      000  DF7:    .BYTE   0,0,0
 5866  050045      000      000      001  DF10:   .BYTE   0,0,1,0,0
 5867  050050      000      000
 5868  050052      000      000      002  DF11:   .BYTE   0,0,2,0,0
 5869  050055      000      000
 5870  050057      000      000      000  DF12:   .BYTE   0,0,0,2,0,0
 5871  050062      002      000      000
 5872  050065      000      000      000  DF13:   .BYTE   0,0,0
 5873  050070      000      000      000  DF16:   .BYTE   0,0,0,0,0,0
 5874  050073      000      000      000
 5875  050076      003      000      000  DF17:   .BYTE   3,0,0,0,0
 5876  050101      000      000
 5877  050103      003      000      000  DF20:   .BYTE   3,0,0,0,0,0,0,0
 5878  050106      000      000      000
 5879  050111      000      000
 5880  050113      000      000      000  DF21:   .BYTE   0,0,0,0
 5881  050116      000
 5882  050117      000      000      000  DF22:   .BYTE   0,0,0,0,0
 5883  050122      000      000
 5884  050124      000      000      000  DF23:   .BYTE   0,0,0
 5885  050127      000      000      000  DF24:   .BYTE   0,0,0,0
 5886  050132      000
 5887  050133      000      000      000  DF30:   .BYTE   0,0,0,0,0,0
 5888  050136      000      000      000
 5889  050141      000      000      000  DF46:   .BYTE   0,0,0,0
 5890  050144      000
 5891  050145      000      000           DF50:   .BYTE   0,0
 5892  050147      000      000      000  DF51:   .BYTE   0,0,0,0
 5893  050152      000
 5894  050153      003      000      000  DF52:   .BYTE   3,0,0,0,0,0,0,0
 5895  050156      000      000      000
 5896  050161      000      000
 5897  050163      000      000      000  DF56:   .BYTE   0,0,0,0
 5898  050166      000
 5899
 5900            000001                            .END
```

```
ABASE = 000000
ACDW1 = 000000
ACDW2 = 000000
ACPUOP= 000000
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
ANDADR  001300
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATESTN= 000000
AUNIT = 000000
AUSWR = 000000
AVECT1= 000000
AVECT2= 000000
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BPTVEC= 000014
CKSWR = 104410
CMPREG  035260
CMSG    036635
CNTRLC  036556
CR    = 000015
CRLF  = 000200
DALTB1  026776
DALTB2  027032
DALTB3  027344
DALTB4  027400
DDISP = 177570
DF1     050022
DF10    050045
DF11    050052
DF12    050057
DF13    050065
DF16    050070
DF17    050076
DF2     050027
DF20    050103
DF21    050113
DF22    050117
DF23    050124
DF24    050127
DF3     050036
DF30    050133
DF46    050141
DF50    050145
DF51    050147
DF52    050153
DF56    050163
DF7     050042
DH1     044005
DH10    044235
DH11    044335
DH12    044455
DH13    044615
DH16    044645
DH17    044745
DH2     044055
DH20    045035
DH21    045163
DH22    045243
DH23    045342
DH24    045372
DH26    045432
DH3     044145
DH30    045472
DH31    045552
DH32    045632
DH33    045712
DH34    045752
DH35    046032
DH36    046112
DH37    046152
DH40    046267
DH42    046327
DH44    046367
DH45    046501
DH46    046676
DH50    046753
DH51    046773
DH52    047033
DH56    047151
DH7     044205
DISPLA  001142
DISPRE  000174
DSWR  = 177570
DT1     047212
DT10    047270
DT11    047304
DT12    047320
DT13    047336
DT16    047346
DT17    047364
DT2     047226
DT20    047400
DT21    047422
DT22    047434
DT23    047450
DT24    047460
DT26    047472
DT3     047246
DT30    047504
DT31    047522
DT32    047540
DT33    047556
DT34    047570
DT35    047606
DT36    047624
DT37    047636
DT40    047654
DT42    047666
DT44    047700
DT45    047716
DT46    047734
DT50    047746
DT51    047754
DT52    047766
DT56    050010
DT7     047260
EMTVEC= 000030
EM1     040714
EM10    041273
EM11    041337
EM12    041377
EM13    041446
EM14    041503
EM15    041536
EM16    041612
EM17    041654
EM2     040754
EM20    041712
EM21    041764
EM22    042021
EM23    042060
EM24    042124
EM25    042164
EM26    042227
EM27    042277
EM3     041023
EM30    042346
EM31    042420
EM32    042465
EM33    042546
EM34    042631
EM37    042707
EM4     041062
EM40    042754
EM41    043023
EM42    043056
EM43    043115
EM44    043164
EM45    043230
EM46    043303
EM47    043346
EM5     041115
EM50    043411
EM51    043447
EM52    043525
EM53    043603
EM54    043646
EM55    043704
EM56    043742
EM6     041170
EM7     041235
ERRTYP  034654
ERRVEC= 000004
FORMPA  035452
GTSWR = 104407
HT    = 000011
IOTVEC= 000020
KERSTK= 001100
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KSP   =%000006
LF    = 000012
LOOP    020504
MGMERR  002150
MGMFLG  002152
MMVEC = 000250
ORADR   001302
PBAHI   001314
PBALO   001312
PDRTB1  027014
PDRTB2  027046
PDRTB3  027362
PDRTB4  027414
PIRQ  = 177772
PIRQVE= 000240
PR0   = 000000
```

```
PR1    = 000040
PR2    = 000100
PR3    = 000140
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
PWRMSG  040652
PWRVEC= 000024
RDCHR  = 104411
RDLIN  = 104412
RESREG= 104414
RESTRT  020462
RESVEC= 000010
R6     =%000006
R7     =%000007
SAVREG= 104413
SETREG  035166
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STACK  = 001100
START   020000
STKLMT= 177774
SWR     001140
SWREG   000176
SW0    = 000001
SW00   = 000001
SW01   = 000002
SW02   = 000004
SW03   = 000010
SW04   = 000020
SW05   = 000040
SW06   = 000100
SW07   = 000200
SW08   = 000400
SW09   = 001000
SW1    = 000002
SW10   = 002000
SW11   = 004000
SW12   = 010000
SW13   = 020000
SW14   = 040000
SW15   = 100000
SW2    = 000004
SW3    = 000010
SW4    = 000020
SW5    = 000040
SW6    = 000100
SW7    = 000200
SW8    = 000400
SW9    = 001000
TBIT   = 000020
TBITPS  001276
TBITVE= 000014
TESTNO  001262
TIMEON  020462
TIMERR  002076
TIMFLG  002100
TIMOFF  033626
TKVEC  = 000060
TOFF    035100
TON     035134
TONUM   001304
TPVEC  = 000064
TRAPPC  001266
TRAPPS  001270
TRAPVE= 000034
TRTVEC= 000014
TST1    020566
TST10   021536
TST11   021670
TST12   022022
TST13   022164
TST14   022222
TST15   022272
TST16   022410
TST17   022532
TST2    020640
TST20   022662
TST21   023012
TST22   023216
TST23   023366
TST24   024116
TST25   024506
TST26   025224
TST27   025420
TST3    020724
TST30   025626
TST31   025774
TST32   026270
TST33   026500
TST34   027062
TST35   027430
TST36   027622
TST37   030224
TST4    021046
TST40   030412
TST41   030474
TST42   030644
TST43   031166
TST44   032034
TST45   032714
TST46   033460
TST47   033564
TST5    021114
TST6    021252
TST7    021404
TYPBN  = 104406
TYPDS  = 104405
TYPE   = 104401
TYPOC  = 104402
TYPON  = 104404
TYPOS  = 104403
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
USESTK= 000700
USP    =%000006
VIRT1   001306
VIRT2   001310
WASR6   001264
WASSR0  001272
WASSR2  001274
WBIT   = 000100
$APTHD  000204
$ATYC   037214
$ATY1   037170
$ATY3   037176
$ATY4   037206
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$BELL   001216
$BIN    037510
$CHARC  037164
$CKSWR  035564
$CLR.T  034046
$CMTAG  001100
$CM1   = 000006
$CM2   = 000014
$CM3   = 000006
$CM4   = 000006
$CNTLC  036514
$CNTLG  036526
$CNTLU  036521
$CPUOP  001254
$CRLF   001223
$DBLK   040154
$DB2O   040260
$DEVCT  001236
$DOAGN  034066
$DTBL   040144
$ENDAD  034056
$ENDCT  033704
$ENULL  034132
$ENV    001246
$ENVM   001247
$EOP    033650
$EOPCT  033676
$ERFLG  001103
$ERMAX  001115
$ERROR  034416
$ERRPC  001116
$ERRTB  001316
$ERTTL  001112
$ESCAP  001214
$ETABL  001246
$ETEND  001262
$FATAL  001230
$FFLG   037434
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  034030
$GTSWR  035634
$HD    = 000000
$HIBTS  000204
$ICNT   001104
$ILLUP  040644
$INTAG  001135
$ITEMB  001114
$LF     001224
$LFLG   037433
$LOOP   034124
$LPADR  001106
$LPERR  001110
$MADR1  001260
$MAIL   001226
$MAMS1  001256
$MBADR  000206
$MFLG   037432
$MNEW   036544
$MSGAD  001242
$MSGLG  001244
$MSGTY  001226
$MSWR   036533
$MTYP1  001257
$MXCNT  034414
$NULL   001154
$NWTST= 000001
$OCNT   037734
$OCTVL  040362
$OMODE  037736
$OVER   034400
$PASS   001234
$PASTM  000212
$PWRAD  040626
$PWRDN  040466
$PWRMG  040622
$PWRUP  040540
$QUES   001222
$RDCHR  036106
$RDLIN  036226
$RDSZ  = 000010
$REGAD  001160
$REG0   001162
$REG1   001164
$REG2   001166
$REG3   001170
$REG4   001172
$REG5   001174
$RESRE  040222
$RTNAD  034126
$RTRN   034122
$SAVRE  040164
$SAVR6  040650
$SCOPE  034136
$SETUP= 000137
$STUP  = 177777
$SVLAD  034344
$SVPC  = 000204
$SWR   = 177400
$SWREG  001250
$SWRMK= 000000
$TBIT   034130
$TESTN  001232
$TIMES  001212
$TKB    001146
$TKS    001144
$TMP0   001176
$TMP1   001200
$TMP2   001202
$TMP3   001204
$TMP4   001206
$TMP5   001210
$TN    = 000050
$TPB    001152
```

```
$TPFLG  001157          $TRPAD  040434          $TYPDS  037740          $TYPON  037552          $XTSTR  034150
$TPS    001150          $TSTM   000210          $TYPE   036706          $TYPOS  037512          $OFILL  037735
$TRAP   040400          $TSTNM  001102          $TYPEC  037120          $UNIT   001240          .       = 050167
$TRAP2  040422          $TTYIN  036504          $TYPEX  037166          $UNITM  000214          .$X     = 000204
$TRP  = 000015          $TYPBN  037436          $TYPOC  037536          $USWR   001252

. ABS.  050167     000


 ERRORS DETECTED:  0

 CJKDAA,CJKDAA/SOL/NL:TOC=CJKDAA.P11
 RUN-TIME: 35 20 .7 SECONDS
 RUN-TIME RATIO: 335/57=5.8
 CORE USED:  32K  (63 PAGES)
```