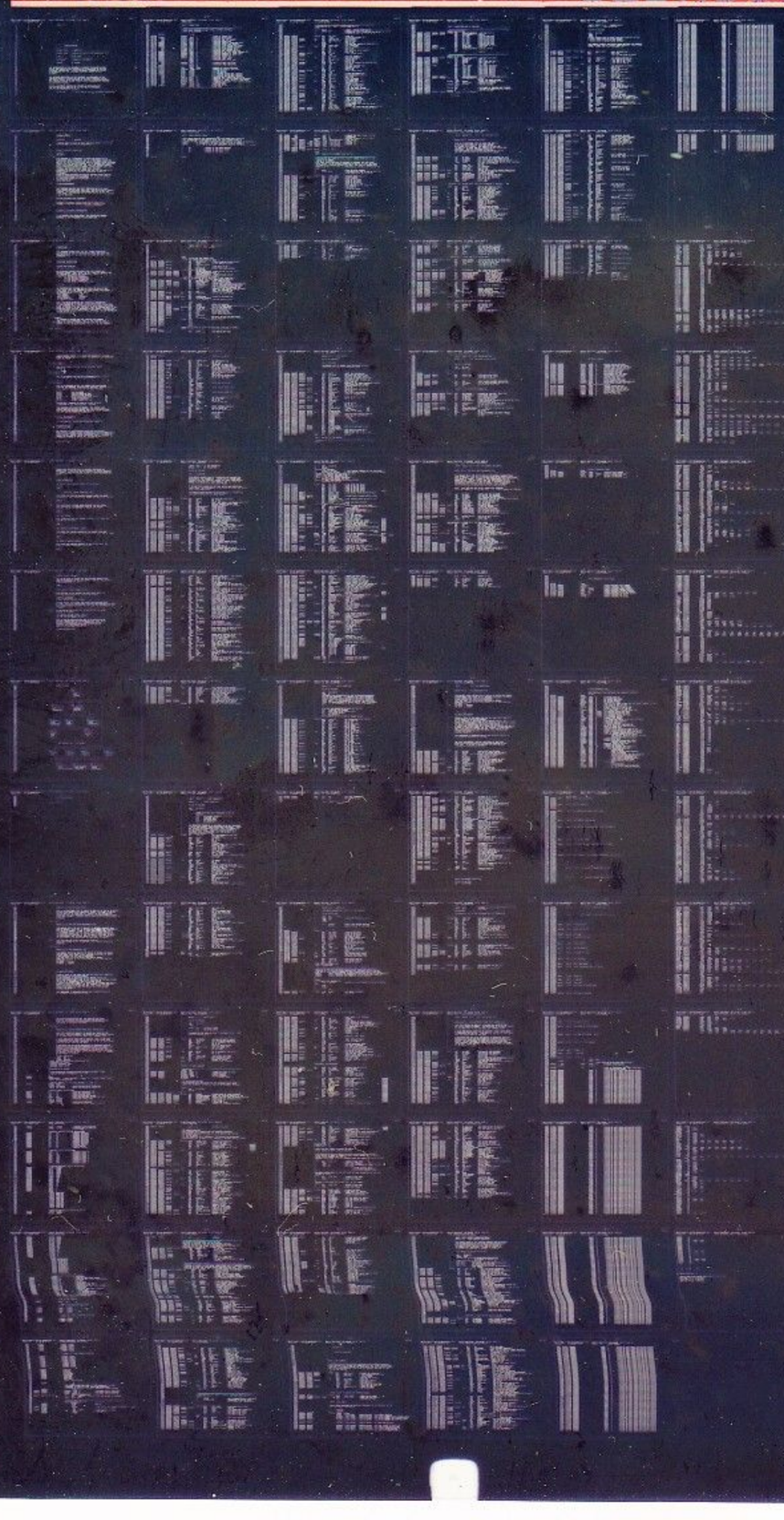
DX11-B

DX-11B ADRSNG TST
CZDXJC0

AH-A830C-MC
FICHE 1 OF 1





```
                .REM    %
```

## IDENTIFICATION

PRODUCT CODE: AC-A829C-MC

PRODUCT NAME: CZDXJC0 DX11-B OFF-LINE ADDRESSING TEST

DATE: MARCH 1982

MAINTAINER: DIAGNOSTIC PROGRAMMING

AUTHOR: R. MISNER

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

                                04-DEC-81       V. GUARINO

                                INSERTED CODE TO FORCE NO MORE THAN 124KW MEMORY ALLOCATION
                                IN THE MEMORY SIZE (SIZE:) SUBROUTINE.



                        1.0     GENERAL PROGRAM INFORMATION

                        1.1     ABSTRACT
                                --------

                                THIS PROGRAM IS A SUPPLEMENT TO DZDXA AND DZDXF DX11-B
                                MAINTENANCE MODE DIAGNOSTICS AND IS A FUNCTIONAL CHECK THE
                                ABILITY OF THE DX TO ACCESS ALL AREAS OF MEMORY USING DIFFERENT
                                SOURCES FOR THE NPR DATA ADDRESS. IT IS ASSUMED THAT DZDXA AND
                                DZDXF HAVE ALREADY BEEN RUN. THE THREE NPR ADDRESS SOURCES
                                RESULT FROM DATA WRITES TO MEMORY, STATUS POINTER WORD (SPW)
                                FETCHES, AND DEVICE STATUS TABLE (DST) FETCHES.
                                DIAGNOSIS IS BASED ON THE ASSUMPTION THAT THE CPU CAN ACCESS
                                MEMORY CORRECTLY WHILE THE DX'S ADDRESSING CAPABILITY
                                IS SUSPECT.
                                ERROR MESSAGES DESCRIBE THE FAILING ADDRESS SOURCE AND THE
                                EXPECTED LOCATION OF THE DX ACCESS. THE ACTUAL DX ADDRESS
                                OF THE ACCESS CANNOT BE DETERMINED.
                                THE OPERATOR INTERFACE IS DESIGNED TO BE COMPATIBLE WITH
                                DZDXA AND DZDXF.

                        1.2     SYSTEM REQUIREMENTS
                                -------------------

                                THIS PROGRAM REQUIRES ANY PDP11 PROCESSOR WITH AT LEAST
                                8K OF MEMORY BUT NOT MORE THAN 128K; THE DX11-B AND A CONSOLE
                                TERMINAL. THE PROGRAM WILL RUN UNDER XXDP,ACT,SLIDE, AND
                                STAND ALONE. THIS PROGRAM HAS NOT BEEN TESTED ON CPU'S WITHOUT
                                A SWITCH REGISTER.

                        1.3     RELATED DOCUMENTS AND STANDARDS
                                -------------------------------

                                PROGRAM IS ASSEMBLED USING SYSMAC MD-11-DZQAC-C3 AND
                                CONFORMS TO THE ACT INTERFACE AUTOCAT-11-QZAUB-B-D
                                                              ------------------



                        1.4     DIAGNOSTIC HIERARCHY PREREQUSITES
                                ---------------------------------

                                IT IS ASSUMED THAT DZDXA,DZDXF, AND APPROPRIATE MEMORY
                                DIAGNOSTICS ARE RUN ERROR FREE IN ORDER FOR THE ERROR
                                MESSAGES IN THIS DIAGNOSTIC TO BE MEANINGFUL.
```

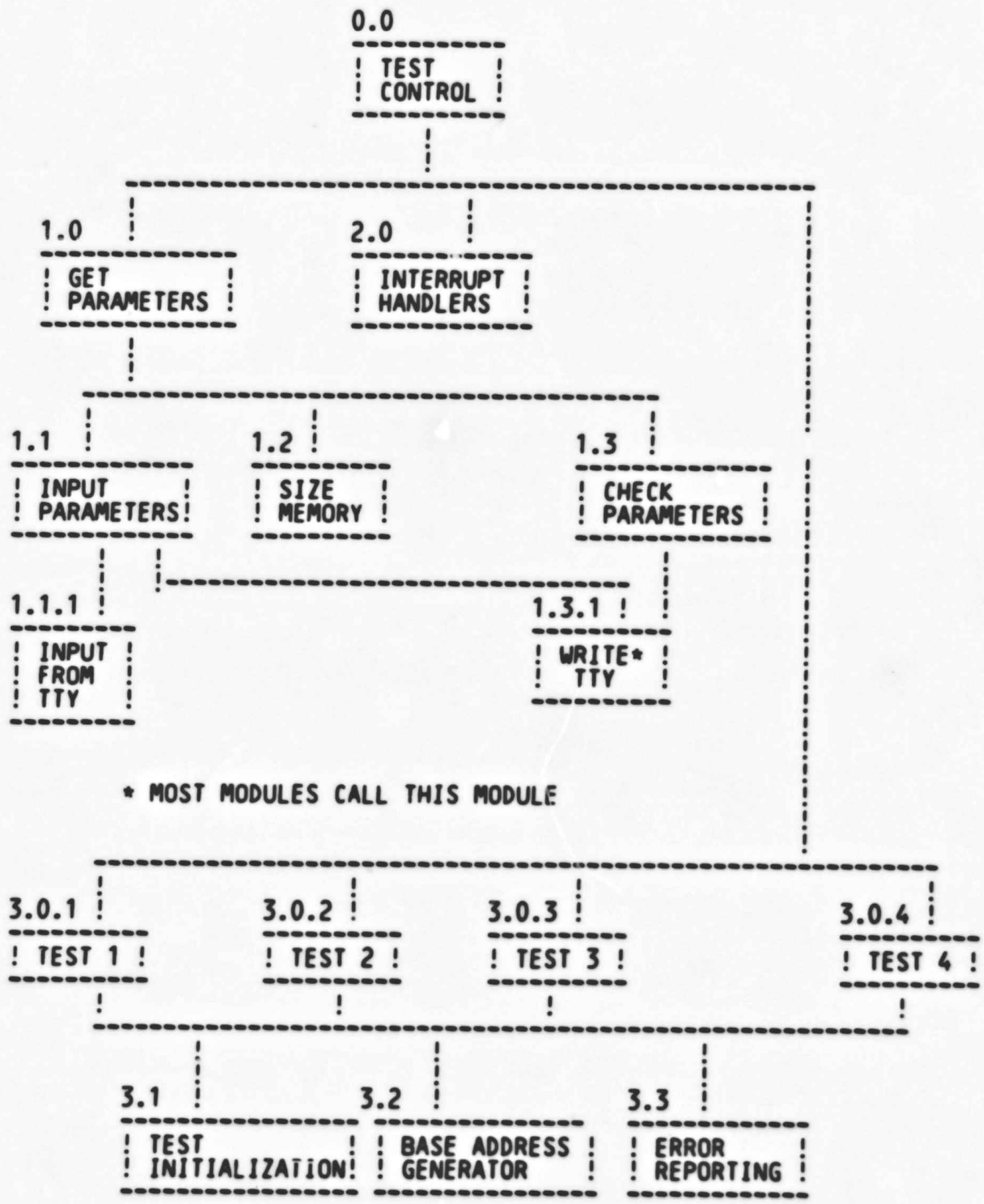
```
                                1.5     ASSUMPTIONS
                                        -----------

                                        THIS PROGRAM ASSUMES THAT THE CPU, UNIBUS ADDRESSING,
                                        AND PORTIONS OF THE DX11-B TESTED BY DZDXA AND DZDXF ARE
                                        OPERATING CORRECTLY IN ORDER FOR THE ERROR MESSAGES TO BE
                                        MEANINGFUL.


                                2.0     OPERATING INSTRUCTIONS
                                        ----------------------

                                2.1     LOADING AND STARTING
                                        --------------------

                                        USE STANDARD LOADING PROCEDURES FOR PDP11 UNDER THE
                                        ABSOLUTE LOADER, XXDP, SLIDE, OR ACT. START AT ADDRESS 200.
                                        THE PROGRAM WILL PRINT OUT THE MAINDEC NUMBER AND TITLE AND
                                        A PROMPTING MESSAGE AS FOLLOWS:
                                        "TYPE:  <D>,FOR DEFAULT PARAMETERS
                                                <P>,FOR PREVIOUS PARAMETERS
                                                <S>,FOR SELECT PARAMETERS
                                                <N>,FOR START WITH THIS TEST NO.

                                        D,P,S,N?"

                                        THE OPERATOR MUST RESPOND WITH A D OR S IF THIS IS THE
                                        FIRST START SINCE THE PROGRAM HAS BEEN LOADED INTO MEMORY. IF,
                                        IN RESPONSE TO THIS MESSAGE, THE OPERATOR TYPES AN S<CR> THE
                                        FOLLOWING DIAGLOG COULD TAKE PLACE:
                                        "TEST NUMBER:               1<CR>
                                         BASE ADDRESS:              176200<CR>
                                         VECTOR ADDRESS:            300<CR>
                                         DX PRIORITY LEVEL:         4<CR>"

                                        NOTE: THE ABOVE RESPONSES ARE THE DEFAULT PARAMETERS
                                        AND ARE EQUIVALENT TO TYPING A D<CR> IN RESPONSE TO D,P,S,N?
                                        A <CR> RESPONSE FOR A PARTICULAR ENTRY WILL SELECT THE DEFAULT
                                        VALUE. A CONTROL C AT ANY TIME WILL TAKE THE PROGRAM BACK TO THE
                                        PARAMETER ENTRY SECTION. RESPONSE TO THE CONTROL C MAY BE SLOW
                                        IF THE PROGRAM IS IN A PARTICULARLY LONG TEST. ALL PARAMETERS ARE
                                        CHECKED TO SEE IF THEY ARE IN A LOGICAL RANGE. IF NOT, THE ENTRY
                                        IS REQUESTED AGAIN.


                        "TEST NUMBER:"
                                THE PROGRAM IS ASKING FOR THE STARTING TEST NUMBER. 1
                                THRU 4 ARE VALID RESPONSES. THIS IS THE STARTING TEST
                                NUMBER. TO LOOP ON A PARTICULAR TEST TYPE THE TEST NUMBER IN
                                RESPONSE TO THIS MESSAGE AND SET SWITCH 14 ON THE CONSOLE.


                        'BASE ADDRESS:"

                                THIS IS A REQUEST FOR THE BASE ADDRESS OF THE DX IN THE
```

```
                        UNIBUS ADDRESS SPACE. IT IS ALSO THE ADDRESS OF THE DXDS. ALL
                        OTHER DX CONTROL AND STATUS REGISTER ADDRESSES ARE GENERATED
                        FROM THIS ENTRY.


                "VECTOR ADDRESS:"

                        THIS IS A REQUEST FOR THE DX11-B INTERRUPT VECTOR
                        ADDRESS.


                "DX PRIORITY LEVEL"

                        THIS IS A REQUEST FOR THE DX'S INTERRUPT PRIORITY LEVEL.
                        IT IS NOT USED BY THIS PARTICULAR PROGRAM.


                "SET SWITCHES:"

                        THIS IS A REMINDER TO SET THE CONSOLE SWITCHES BEFORE
                        STARTING THE PROGRAM. AFTER SETTING SWITCHES, RESPOND WITH A
                        <CR>. CONSOLE SWITCHES HAVE THE FOLLOWING MEANING:

                        SW<15>          HALT ON ERROR
                        SW<14>          LOOP ON ERROR OR TEST
                        SW<13>          INHIBIT ERROR PRINTOUT
                        SW<12>          PRINT ONLY SHORT ERROR REPORT
                        SW<11>          INHIBIT ITERATIONS
                        SW<10>          INHIBIT MEMORY MGT USE

                        TESTING WILL COMMENCE AS SOON AS A <CR> IS GIVEN
                        IN RESPONSE TO THE "SET SWITCHES:" MESSAGE.

                        THIS PROGRAM AUTO SIZES THE MEMORY. THE CONTROL UNIT
                        ADDRESS AND DEVICES PER CONTROL UNIT ARE ALSO AUTO SIZED.
                        THIS PARTICULAR PROGRAM DOES NOT NEED THE "LEGAL COMMANDS"
                        FACILITY USED BY OTHER DX DIAGNOSTICS.


        2.2     SPECIAL ENVIRONMENTS
                ----------------------

                        THIS PROGRAM WILL RUN IN UNATTENDED MODE UNDER ACT.


        2.3     PROGRAM OPTIONS
                ---------------

                        SEE SECTION 2.1 FOR PARAMETER SELECTION AND OPTIONS.
                        NOTE THAT ALL CONSOLE SWITCH CHANGES ARE EFFECTIVE IMMEDIATELY
                        EXCEPT SW<10> "INHIBIT MEMORY MANAGEMENT USES". IN ORDER TO
                        CHANGE THE MEANING OF SW<10> SET THE SWITCH THEN RESTART THE
                        PROGRAM VIA CONTROL C OR START AT ADDRESS 200.


        2.4     EXECUTION TIMES
```

```
                                        ---------------

                                        SINGLE PASS EXECUTION TIME IS VARIABLE DEPENDING ON
                                        MEMORY SIZE. A PDP-11/45 WITH 65K WORDS OF MEMORY REQUIRES
                                        50 SECONDS. EACH ROUTINE HAS A SINGLE ITERATION REGARDLESS
                                        OF MODE. END PASS IS SIGNALED AFTER ALL TESTS HAVE BEEN RUN
                                        ONCE.


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

                                        THE GENERAL ERROR REPORT FORMAT IS AS FOLLOWS:

                                        LINE 1
                                        ------

                                                ERROR PC: XXXXXX ERROR NO. YYYYY

                                        THIS GIVES THE PROGRAM COUNTER (XXXXXX) WHERE THE ERROR
                                        WAS DETECTED AND A UNIQUE ERROR NUMBER OR IDENTIFIER (YYYYYY)

                                        LINE 2
                                        ------

                                        THIS LINE GIVES A UNIQUE ERROR MESSAGE FOR EACH ERROR
                                        TYPE EXPLAINING THE SUSPECTED CAUSE OF THE FAILURE. THIS IS
                                        THE LAST LINE OF THE MESSAGE IF SWITCH 12 IS ON.

                                        LINE 3
                                        ------

                                        TEST NO. XXXXXX

                                        THIS IDENTIFIES THE TEST NUMBER (XXXXXX = 1-4) WHERE THE
                                        FAILURE WAS DETECTED.

                                        LINE 4
                                        ------

                                        VIRT ADDR OF BUFFER BASE: XXXXXX

                                        THIS GIVES THE VIRTUAL ADDRESS OF THE LOWEST LOCATION
                                        IN THE BUFFER WHICH THE DX WAS ACCESSING AT THE TIME OF ERROR.

                                        LINE 5
                                        ------

                                        PHY ADDR OF BUFFER: XXXXXX

                                        THIS GIVES THE PHYSICAL ADDRESS OF THE BUFFER BASE. ONLY
                                        OUTPUT IF MEMORY MANAGEMENT IS IN USE

                                        LINE 6
                                        ------
```

```
                        EA BITS IN OCT: XXXXXX

                        THE LOW TO BITS OF XXXXXX ARE THE EXTENDED ADDRESS BITS
                        USED IN ACCESSING THE FAILING MEMORY LOCATION. ONLY OUTPUT IF
                        MEMORY MANAGEMENT IN USE.

                        LINE 7
                        ------

                        VIRT ADDR OF ACCESS ERROR: XXXXXX

                        THIS MESSAGE IS ONLY OUTPUT FOR TEST 2. IT GIVES THE
                        EXPECTED WORD LOCATION OF THE FAILING ACCESS.

                        LINE 7-8 ALTERNATE
                        ACTUAL: XXXXX EXPECTED: XXXXXX

                        IN SOME TESTS, THIS GIVES THE ACTUAL AND EXPECTED BYTE
                        OF DATA. THIS WILL GIVE INSIGHT INTO FAILING ADDRESSES AS THERE
                        IS A CORRESPONDENCE BETWEEN THIS DATA AND THE LOCATION OF THE
                        FAILURE.

                NOTE:   AT THE END OF THE FIRST PASS EXECUTION THE FOLLOWING
                        MESSAGE IS PRINTED:

                        1ST IBM DEV RCGNZD. OCTAL: XXXXXX
                        SIZE OF 1ST DEV GROUP: YYYYYY

                        THIS IS NOT AN ERROR MESSAGE BUT IS AN INDICATION OF
                        HOW THE JUMPERS ARE SET UP ON MODULE A20. THIS INFOR-
                        MATION IS PROVIDED ONLY AS A CONVENIENCE FOR THE
                        OPERATOR. SEE MAINTENANCE MANUAL EK-DX11B-MM-002
                        PARA. 2.4.2 FOR MORE INFORMATION.

            4.0         PROGRAM ORGANIZATION
```

## 4.1 BLOCK DIAGRAM



```
362
363                             4.2     TEST DESCRIPTIONS
364
365
366                             4.2.1 TST 1
367
368
```

```
                                        MODULE 3.0.1 FAST NPR DATA TEST

                                        THIS IS A FUNCTIONAL TEST TO CHECK THE ABILITY OF THE DX TO DO NPR'S
                                        OF DATA INTO MEMORY WITH LARGE BYTE COUNTS. THE DXBA IS CHECKED
                                        FOR ITS ABILITY TO COUNT AND ALL CARRIES IN THE COUNTER ARE
                                        EXERCISED. THE DATA TRANSFERS ACCESS ALL AVAILABLE MEMORY EXCEPT
                                        WHERE THE PROGRAM IS LOCATED, AND THE AREA WHICH MAY BE OCCUPIED BY
                                        A LOADER.

                                4.2.2 TST 2

                                        MODULE 3.0.2 NPR DATA ADDR UNIQUENESS TEST

                                        THIS MODULE TESTS THE DX11'S ABILITY TO UNIQUELY ADDRESS EACH
                                        WORD OF MEMORY. THIS TEST IS ACCOMPLISHED BY WRITING A BACKGROUND
                                        IN A BLOCK OF MEMORY. A WORD IS THEN TRANSFERRED FROM THE DX
                                        TO THE MEMORY BLOCK. THE CPU CHECKS THAT THE WORD ARRIVES AT
                                        ITS EXPECTED DESTINATION. IF YES THE WORD IS RESTORED TO MATCH
                                        THE BACKGROUND, AND THE NEXT WORD IS TESTED. THIS IS REPEATED
                                        FOR EACH WORD IN THE BLOCK AND FOR EACH BLOCK IN MEMORY.

                                NOTE:   THIS TEST MAY NOT DETECT ADDRESSING PROBLEMS
                                        WHICH ARE COMMON TO THE CPU AND DX11. I.E. UNIBUS ADDRESSING
                                        PROBLEMS.

                                4.2.3 TST 3

                                        MODULE 3.0.3 SPW ADDRESSING TEST

                                        THIS MODULE TESTS THE ABILITY OF THE DX TO ACCESS THE STATUS
                                        POINTER WORD (SPW) TABLE IN ALL POSSIBLE LOCATIONS IN MEMORY.
                                        A BASE ADDRESS FOR THE SPW IS GENERATED.
                                        THIS SPW TABLE IS FILLED WITH A PATTERN OF 000DEV
                                        WHERE DEV IS EQUAL TO THE DISPLACEMENT OF THE WORD FROM THE
                                        SPW BASE; I.E.
                                        SPW BASE     =000000
                                        SPW BASE +1  =000001
                                                  ..
                                                  ..
                                                  ..
                                                  ..
                                        SPW BASE+255=000377

                                        GIVING A TOTAL OF 256 WORDS FOR THE SPW TABLE WITH IMMEDIATE
                                        STATUS EQUAL TO THE DEVICE NUMBER.
                                        AN ISS SEQUENCE IS INITIATED WITH A DEVICE # OF 0. IF
                                        ADRECC AND ADRECD DON'T COME ON FOR THIS DEV, NO TESTING IS DONE,
                                        THE DEV # IS INCREMENTED, AND ISS IS DONE AGAIN.
                                        IF ADRECC AND ADRECD COME ON IT MEANS THE DX JUMPERS ARE CUT TO
                                        RESPOND TO THIS DEV NUMBER. THE ISS SEQUENCE IS THEN CONTINUED
                                        UNITL THE STATUS IS IN THE DXOS REG. THIS STATUS IS THEN
                                        CHECKED AND SHOULD BE EQUAL TO THE DEV # IF THE SPW WAS
                                        ACCESSED CORRECTLY.

                                        THIS PROCEDURE IS REPEATED FOR ALL 256 POSSIBLE DEVICES AT
                                        ALL POSSIBLE LOCATIONS FOR THE SPW.
```

```
                                THE FIRST RECOGNIZED DEV NO. AND COUNT OF THE FIRST
                                GROUP OF RECOGNIZED DEVICES IS SAVED AND PRINTED AT END
                                PASS TIME. IF NO DEV IS RECOGNIZED AND ERROR MESSAGE
                                IS OUTPUT AT THE END OF THE TEST.

                        4.2.4 TST 4

                                MODULE 3.0.4 DEVICE STATUS TABLE (DST) ACCESS TEST

                                THIS TEST CHECKS THE ABILITY TO ACCESS ALL BYTES OF THE
                                DST IN ALL AREAS OF MEMORY. THE OBJECTIVE IS TO CHECK LOADING
                                OF THE DXBA FROM THE SPW DST BASE, AND  COMMAND (CUCR)

                                A VALID IBM DEVICE FROM TST3 IS CHECKED. THEN A BASE ADDR
                                IS GENERATED FOR THE DST. EACH BYTE OF THE DST IS THEN LOADED
                                WITH ITS OFFSET FROM THE DST BASE.

                                A SPW ENTRY FOR THE VALID IBM DEVICE IS GENERATED WITH AP-
                                PROPRIATE DST BASE AND ZERO IMMEDIATE STATUS TO ASSURE A DST
                                ACCESS. AN ISS SEQUENCE IS DONE WITH THE VALID DEV AND
                                EVERY POSSIBLE IBM COMMAND THEREBY ACCESSING EVERY LOCATION
                                IN THE DST. AFTER THE ISS, THE STATUS WILL EQUAL THE COMMMAND
                                OR AND ERROR IS FLAGGED.
                                %
                                .NLIST  CND,MC,TOC
                                .LIST   ME
                                .ENABL  ABS,AMA
                        .TITLE  MD-11-CZDXJ-C DX11-B ADDRESSING TEST
                        ;*COPYRIGHT (C) -1977-
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY R. MISNER
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                        ;*
000001                  $TN=1
160000                  $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
                        .SBTTL  BASIC DEFINITIONS

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

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

```
                        177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
                        177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

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

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

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

                                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                                  BIT15=  100000
040000                                  BIT14=  40000
```

```
                        020000          BIT13=  20000
                        010000          BIT12=  10000
                        004000          BIT11=  4000
                        002000          BIT10=  2000
                        001000          BIT09=  1000
                        000400          BIT08=  400
                        000200          BIT07=  200
                        000100          BIT06=  100
                        000040          BIT05=  40
                        000020          BIT04=  20
                        000010          BIT03=  10
                        000004          BIT02=  4
                        000002          BIT01=  2
                        000001          BIT00=  1
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
                                        .EQUIV  BIT01,BIT1
                                        .EQUIV  BIT00,BIT0

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

                                        ;*KT11 VECTOR ADDRESS

                        000250          MMVEC=  250

                                        ;*KT11 STATUS REGISTER ADDRESSES

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

                                        ;*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

                    000200          DONE=200
                    000004          SOSIEN=4
                    020000          SELO=20000
                    040000          HLDO=40000
                    002000          CMDO=2000
                    001000          SRVO=1000
                    004000          ADRO=4000
                                    .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  001164                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
                                    .SBTTL  ACT11 HOOKS

                                    ;;**********************************************************************
                                    ;HOOKS REQUIRED BY ACT11
                    000204                  $SVPC=.                 ;SAVE PC
                    000046                  .=46
000046  001566                      $ENDAD                          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                    000052                  .=52
000052  040000                      .WORD   40000                   ;;2)SET LOC.52 TO 40000
                    000204                  .=$SVPC                 ;; RESTORE PC
                    001000                  .=1000
```

```
                                .SBTTL  COMMON TAGS

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

001100                                  .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP 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  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001161  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001162  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
```

```
                                .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


001164                          $ERRTB:
```

```
                                        .SBTTL  MODULE 0.0      TEST CONTROL
                                ;       MODULE 0.0      TEST CONTROL
                                ;
                                ;
                                ;
                                ;
                                ;
001164                          START:
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001164  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001170  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001172  022706  001140                  CMP     #SWR,R6 ;;DONE?
001176  001374                          BNE     .-6             ;;LOOP BACK IF NO
001200  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001204  012737  006174  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001212  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001220  012737  006272  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001226  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001234  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001240  012737  001274  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001246  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001254  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001262  022777  177777  177650          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001270  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001272  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
001274  012716  001302          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001300  000002                          RTI
001302  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001310  012737  000174  001142          MOV     #DISPREG,DISPLAY
001316  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
001322  000005                          RESET
001324  005037  012174                  CLR     FIRSTP          ;CLEAR 1ST PASS SW
001330  005037  012274                  CLR     DEVMSS          ;SET UP TO PRINT DEVICE ADDR CONFIG FIRST TIME ONLY
001334  004737  001600          RESTART:JSR     PC,GETPRM       ;TALK TO THE OPERATOR
001340  013737  012212  012214          MOV     COUNT,CNT       ;SET UP ITERATION COUNT
001346  013737  012154  012206  GO:     MOV     ATESTN,CTESTN   ;GET FIRST TEST NO.
001354  005037  012220                  CLR     LOCK            ;CLEAR LOOP LOCK SW
001360  005037  012216                  CLR     ERR             ;RESET ERROR INDICATOR
001364  013702  012206          TST:    MOV     CTESTN,R2
001370  006302                          ASL     R2              ;MULTIPLY BY TWO FOR WORD BNDRY
001372  000172  001376                  JMP     @TSTAB(R2)      ;GO TO TEST THRU DISPATCH TABLE
001376  000000          TSTAB:  HALT                    ;TEST 0 DOESN'T EXIST
001400  006454                  TST1
001402  007176                  TST2
001404  007444                  TST3
001406  010156                  TST4
                                ;WHEN TEST IS DONE IT ALWAYS RETURNS HERE AFTER ONE ITERATION
001410                          ENDTST:
001410  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
001412  012746  001420                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
```

```
                                                    RTI                         ;;POP NEW PC AND PS
                                            64$:
                                                    BIT     #40000,@SWR         ;IS LOOP SWITCH ON
                                                    BNE     TST                 ;BR IF YES
                                                    INC     CTESTN              ;INCREMENT TO NEXT TEST
                                                    CMP     CTESTN,HTESTN       ;HAVE ALL TEST BEEN RUN
                                                    BLE     TST                 ;BR IF NO, DISPATCH TO NEXT ONE
                                                    DEC     CNT                 ;DECREMENT ITERATION COUNT
                                                    BNE     GO                  ;START OVER IF NOT EXPIRED
                                            EOP:    TST     DEVMSS              ;HAVE WE PRINTED DEVICE CONFIG YET?
                                                    BNE     10$                 ;BR IF YES
                                                    TST     DEVCT               ;WERE DEVICES RECOGNIZED?
                                                    BEQ     10$                 ;BR IF NO, NOTHING TO SAY
                                                    JSR     R5,PRINT            ;PRINT DEV NO. ASCII
                                                    .WORD   DEVMS
                                                    .WORD   0
                                                    JSR     R5,PRINT            ;PRINT DEV NO. OCTAL
                                                    .WORD   FRSTDV
                                                    .WORD   100001
                                                    JSR     R5,PRINT            ;PRINT DEVICE COUNT ASCII
                                                    .WORD   DEVCMS
                                                    .WORD   0
                                                    JSR     R5,PRINT            ;PRINT DEVICE COUNT OCT
                                                    .WORD   DEVCT
                                                    .WORD   100001
                                                    MOV     #1,DEVMSS           ;SET DEV CONFIG PRINTED SWITCH
                                            10$:    TST     KT                  ;IS KT IN USE?
                                                    BEQ     15$                 ;BR IF NO
                                                    CLR     @#SR0               ;TURN OFF MEM MGT
                                            15$:    JSR     R5,PRINT
                                                    .WORD   BELL
                                                    .WORD   0                   ;PRINT CONTROL PARAMETER
                                                    MOV     @#42,R0             ;ARE WE UNDER ACT
                                                    BEQ     END1                ;BR IF NO, START OVER
                                                    RESET
                                            $ENDAD: JSR     PC,@R0              ;GO TO MONITOR
                                                    NOP                         ;ROOM FOR
                                                    NOP                         ;    ACT11
                                                    NOP                         ;    STUFF
                                            END1:   BR      RESTART             ;START OVER
```

```
                                        .SBTTL  MODULE 1.0  GET PARAMETERS
                                ;       MODULE  1.0     GET PARAMETERS
                                ;
                                ;       CALLED AS FOLLOWS
                                ;
                                ;       JSR     PC,GETPRM
                                ;
                                ;       THIS MODULE ASSURES THAT AL  PARAMETERS ARE AQUIRED FOR
                                ;       OPERATION OF THE DIAGNOSTI(   TO DO THIS IT ESTABLISHES
                                ;       A DIALOG WITH THE OPERATOR THROUGH THE INPUT PARAM MODULE.
                                ;       MEMORY IS SIZED BY THE 'SIZE' MODULE, AND ALL PARMS ARE
                                ;       CHECKED FOR VALIDITY BY THE CHECKP MODULE.
                                ;
                                ;       ^C AT ANY TIME CLEARS THE 1ST PASS SWITCH AND EVERYTHING
                                ;       STARTS OVER.  CR IN RESPONSE TO ANY QUESTION CAUSES THE DEFAULT
                                ;       VALUE TO BE SUBSTITUTED.
                                ;
                                ;       THIS MODULE NEVER RETURNS TO CALLER UNLESS ALL PARAMETERS
                                ;       HAVE BEEN ESTABLISHED WITH GOOD VALUES.
                                ;
001600  005737  012174          GETPRM: TST     FIRSTP          ;IS THIS THE FIRST PASS
001604  001402                          BEQ     5$              ;BR IF YES
001606  000137  002270                  JMP     ENDPRM          ;DON'T GET PARM IF WE ALREADY HAVE THEM
001612  005777  176224          5$:     TST     @42             ;ARE WE RUNNING IN AUTO MODE
001616  001415                          BEQ     10$             ;BRANCH IF MACHINE IS ATTENDED
001620  013737  012124  012154          MOV     DTESTN,ATESTN   ;SETUP DEFAULTS
001626  013737  012126  012156          MOV     DUBA,AUBA       ;SETUP DEFAULTS
001634  013737  012130  012160          MOV     DVECT,AVECT     ;SETUP DEFAULTS
001642  013737  012132  012162          MOV     DPRIOR,APRIOR   ;SETUP DEFAULTS
001650  000564                          BR      90$             ;GO SEE ABOUT MEMORY SIZING
001652  022737  001566  000042  10$:    CMP     #$ENDAD,@#42    ;UNDER ACT11?
001660  001560                          BEQ     90$             ;NO OPERATOR INTERACTION
001662  005737  012210                  TST     PARMV           ;ARE PARAMETERS GOOD
001666  001004                          BNE     15$             ;BRANCH IF YES BECAUSE THIS
                                                                ;ALSO MEANS HEADER ISN'T NEEDED
001670  004537  005372                  JSR     R5,PRINT        ;CALL THE PRINTER
001674  012606                          HEADER                  ;ADDRESS OF HEADER ASCIZ
001676  000000                          .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
001700  004537  002272          15$:    JSR     R5,INPRM        ;GET D,P,S, OR N FROM OPER.
001704  000001                          .WORD   1               ;THIS TELLS INPRM TO GET D,P,S,N
                                        ;INPRM WILL RETURN IN R0 THE CHAR TYPED
001706  122700  000104                  CMPB    #'D,R0          ;WAS DEFAULT PARMS REQUESTED?
001712  001015                          BNE     20$             ;BR IF NO
001714  013737  012124  012154          MOV     DTESTN,ATESTN   ;DEFAULT SETUP
001722  013737  012126  012156          MOV     DUBA,AUBA       ;DEFAULT SETUP
001730  013737  012130  012160          MOV     DVECT,AVECT     ;DEFAULT SETUP
001736  013737  012132  012162          MOV     DPRIOR,APRIOR   ;DEFAULT SETUP
001744  000526                          BR      90$             ;GO SEE ABOUT MEMORY SIZING
001746  122700  000120          20$:    CMPB    #'P,R0          ;DOES OPERATOR WANT PREVIOUS
001752  001011                          BNE     30$             ;BR IF NO
001754  005737  012210                  TST     PARMV           ;ARE PARMS SET UP
001760  001401                          BEQ     25$             ;BR IF PARAMETERS NOT VALID
001762  000517                          BR      90$             ;USE PREVIOUS PARMS, NO ACTION NEEDED
001764  004537  005372          25$:    JSR     R5,PRINT        ;TROUBLE, P TYPED AND PARMS
001770  013044                          .WORD   INVM            ;NEVER SET UP PROPERLY
```

```
849  001772  000000                        .WORD   0           ;CONTROL WORD,CRLF + DONT CONVERT
850  001774  000726                        BR      10$         ;GO TO TRY AGAIN
851  001776  122700  000123    30$:       CMPB    #'S,R0      ;ARE PARMS TO BE SELECTED?
852  002002  001062                        BNE     40$         ;BR IF NO
853  002004  005037  012210    35$:       CLR     PARMV       ;CLEAR PARM VALID SW
854  002010  004537  002272                JSR     R5,INPRM    ;ASK INPUT MODULE TO GET TEST NO:
855  002014  000002                        .WORD   2           ;TEST NUMBER WILL BE RETURNED IN R0
856                                                            ;IN OCTAL
857  002016  010037  012154                MOV     R0,ATESTN   ;SAVE TEST NUMBER IN ACTIVE PARM TABLE
858  002022  004537  005224                JSR     R5,CHECKP   ;CHECK IF TEST NO OK
859  002026  000002                        .WORD   2           ;PARAMETER INDICATES VERIFY TEST #
860  002030  005700                        TST     R0          ;R0=0 IF CHECK OK
861  002032  001364                        BNE     35$         ;STAY HERE UNTIL OPERATOR GETS IT RIGHT
862  002034  004537  002272    36$:       JSR     R5,INPRM    ;ASK INPUT MODULE TO GET BASE ADDR
863  002040  000003                        .WORD   3           ;PARAMETER INDICATING GET UBA BASE
864  002042  010037  012156                MOV     R0,AUBA     ;SAVE UB BASE ADDR IN ACTIVE TABLE
865  002046  004537  005224                JSR     R5,CHECKP   ;CHECK FOR VALID ENTRY
866  002052  000003                        .WORD   3           ;PARM TO INDICATE CHECK UBA
867  002054  005700                        TST     R0          ;R0=0 IF CHECK OK
868  002056  001366                        BNE     36$         ;STAY HERE UNTIL HE GETS IT RIGHT
869  002060  004537  002272    37$:       JSR     R5,INPRM    ;ASK INPUT MODULE TO GET VECTOR ADDR
870  002064  000004                        .WORD   4           ;PARM INDICATING VECTOR NEEDED
871  002066  010037  012160                MOV     R0,AVECT    ;SAVE VECTOR IN ACTIVE PARM TABLE
872  002072  004537  005224                JSR     R5,CHECKP   ;CHECK FOR VALID VECTOR
873  002076  000004                        .WORD   4           ;PARM INDICATING CHECK VECTOR
874  002100  005700                        TST     R0          ;R0=0 IF VECTOR OK
875  002102  001366                        BNE     37$         ;STAY HERE UNTIL HE GETS IT RIGHT
876  002104  004537  002272    38$:       JSR     R5,INPRM    ;GO GET DX PRIORITY
877  002110  000005                        .WORD   5
878  002112  010037  012162                MOV     R0,APRIOR   ;SAVE PRIORITY
879  002116  004537  005224                JSR     R5,CHECKP   ;CHECK PRIORITY ENTRY
880  002122  000005                        .WORD   5           ;PARM INDICATING PRIORITY CHECK
881  002124  005700                        TST     R0
882  002126  001366                        BNE     38$         ;STAY HERE UNTIL HE GETS IT RIGHT
883  002130  004537  005372                JSR     R5,PRINT    ;PRINT SET SWITCHES MESSAGE
884  002134  013220                        .WORD   SETSW       ;ADDR OF MESSAGE
885  002136  000000                        .WORD   0
886  002140  004537  002532                JSR     R5,INASC    ;DUMMY INPUT REQUEST TO WAIT FOR
887  002144  000001                        .WORD   1           ;OPERATOR TO SET SWITCHES
888  002146  000425                        BR      90$         ;GO SEE ABOUT MEMORY SIZING
889  002150  122700  000116    40$:       CMPB    #'N,R0      ;WAS AN 'N' INPUT
890  002154  001236                        BNE     10$         ;BR IF NONE OF DPSN WERE INPUT
891  002156  004537  002272    45$:       JSR     R5,INPRM    ;CALL FOR TEST NUMBER TO BE INPUT
892  002162  000002                        .WORD   2           ;PARM SAYS GET TEST #.
893  002164  010037  012154                MOV     R0,ATESTN   ;SAVE STARTING TEST #
894  002170  004537  005224                JSR     R5,CHECKP   ;HAVE TEST # CHECKED FOR PROPER
895  002174  000002                        .WORD   2           ;RANGE
896  002176  005700                        TST     R0          ;IS TEST # OK?
897  002200  001366                        BNE     45$         ;STAY HERE UNTIL HE GETS IT RIGHT
898  002202  005737  012210                TST     PARMV       ;WE HAVE STARTING TEST #, ARE OTHER PARMS OK?
899  002206  001005                        BNE     90$         ;BR IF OK
900  002210  004537  005372                JSR     R5,PRINT    ;PRINT
901  002214  013062                        INVPRM              ;POINTER TO "NEED MORE PARMS" MSG
902  002216  000000                        .WORD   0           ;CONTROL WORD, CRLF + DON'T CONVERT
903  002220  000614                        BR      10$         ;GO BACK AND DO IT ALL AGAIN
904                              ;
```

```
   905
   906  002222  004737  004224          90$:    JSR     PC,SIZE          ;CALL MEMORY SIZING MODULE
   907  002226  012700  000015                  MOV     #15,R0   ;START OF LOOP TO SET UP DX ACCESS ADDRESSES
   908  002232  012701  012054                  MOV     #DXDSA,R1        ;GET ADDR OF START OF TABLE
   909  002236  013702  012156                  MOV     AUBA,R2          ;GET DX REG BASE ADDRESS
   910  002242  010221                  95$:    MOV     R2,(R1)+         ;MOVE UBA TO TABLE
   911  002244  062702  000002                  ADD     #2,R2            ;STEP TO NEXT UBA
   912  002250  005300                          DEC     R0               ;DECREMENT  COUNTER
   913  002252  001373                          BNE     95$              ;FILL THE 13 LOCATIONS OF THE TABLE
   914  002254  012737  000001  012210          MOV     #1,PARMV         ;SET PARAMETER VALID SWITCH
   915  002262  012737  000001  012174          MOV     #1,FIRSTP        ;SET NOT FIRST PASS SWITCH
   916  002270  000207                  ENDPRM: RTS     PC               ;RETURN TO CALLER, END OF MODULE
```

```
                                        .SBTTL  MODULE 1.1  INPUT PARAMETERS

                                ;       MODULE 1.1      INPUT PARAMETERS
                                ;
                                ;       CALLED AS FOLLOWS
                                ;
                                ;       JSR     R5,INPRM
                                ;       .WORD   X
                                ;
                                ;               WHERE X DEFINES WHAT PARAMETER TO GET AS
                                ;               FOLLOWS:
                                ;               X=1     D,P,S,ORN  PARAMETER SELECTION
                                ;               X=2     STARTING TEST #
                                ;               X=3     UBA BASE ADDR
                                ;               X=4     DX VECTOR ADDR
                                ;               X=5     DX PRIORITY
                                ;
                                ;       THE REQUESTED PARAMETER IS RETURNED IN R0 ALL CONVERTED
                                ;       TO OCTAL AND DEFAULT SUBSTITUTED WHEN INDICATED.  INPUTS
                                ;       ARE CHECKED FOR EVERYTHING EXCEPT THAT THE NUMBERS ARE
                                ;       IN THE CORRECT RANGE.  CHECKING INCLUDES NUMERIC IF
                                ;       REQUIRED, OCTAL IF REQUIRED, <CR> FOR DEFAULT, ETC.
                                ;       THIS MODULE DESTROYS R1,R0
002272  012501                  INPRM:  MOV     (R5)+,R1        ;GET THE PARAMETER FROM THE
                                                                ;CALLING CODE.
002274  120127  000001                  CMPB    R1,#1           ;IS PARM 1 - GET D,P,S, ORN?
002300  001016                          BNE     10$             ;BR IF NO.
002302  004537  005372                  JSR     R5,PRINT        ;PRINT
002306  013102                          .WORD   PROMPT          ;D,P,S,N?
002310  000000                          .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
002312  004537  002532                  JSR     R5,INASC        ;READ THE INPUT -
002316  000001                          .WORD   1               ;INASC RETURNS POINTER TO
002320  111100                          MOVB    (R1),R0         ;INPUT IN R1
                                                                ;1ST CHAR TO R0
002322  120027  000015                  CMPB    R0,#CR          ;WAS IT DEFAULTLED?
002326  001100                          BNE     90$             ;NO, ALL DONE
002330  113700  000120                  MOVB    'P,R0           ;INSERT DEFAULT OF P=PREVIOUS PARMS.
002334  000475                          BR      90$             ;ALL DONE

002336  120127  000002          10$:    CMPB    R1,#2           ;IS PARM 2 - GET TEST #
002342  001014                          BNE     20$             ;BR IF NO
002344  004537  005372                  JSR     R5,PRINT        ;PRINT
002350  013114                          .WORD   TESTN           ;ADDR OF STARTING TESTN MSG.
002352  000000                          .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
002354  004537  002532                  JSR     R5,INASC        ;GET OCTAL INPUT
002360  000002                          .WORD   2               ;PARM TO GET OCTAL INPUT
002362  011100                          MOV     (R1),R0         ;GET INPUT #
002364  001061                          BNE     90$             ;IF NON ZERO, WE'RE DONE
002366  013700  012124                  MOV     DTESTN,R0       ;SUBSTITUTE THE DEFAULT
002372  000456                          BR      90$             ;DONE
002374  120127  000003          20$:    CMPB    R1,#3           ;WAS UBA REQUESTED
002400  001014                          BNE     30$             ;BR IF NO
002402  004537  005372                  JSR     R5,PRINT        ;PRINT
002406  013137                          .WORD   BASEA           ;ASK FOR BASE ADDR
002410  000000                          .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
```

```
  973  002412  004537  002532          JSR     R5,INASC        ;GET THE OCTAL
  974  002416  000002                  .WORD   2               ;INPUT
  975  002420  011100                  MOV     (R1),R0         ;GET THE NUMBER INPUT
  976  002422  001042                  BNE     90$             ;BR IF NOT DEFAULTED
  977  002424  013700  012126          MOV     DUBA,R0         ;SUBSTITUTE THE DEFAULT
  978  002430  000437                  BR      90$             ;DONE
  979  002432  120127  000004  30$:    CMPB    R1,#4           ;DOES CALLER WANT VECTOR?
  980  002436  001014                  BNE     40$             ;BR IF NO
  981  002440  004537  005372          JSR     R5,PRINT        ;PRINT
  982  002444  013155                  .WORD   VECTA           ;ASK FOR VECTOR ADDR
  983  002446  000000                  .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
  984  002450  004537  002532          JSR     R5,INASC        ;GET THE INPUT
  985  002454  000002                  .WORD   2               ;IN OCTAL
  986  002456  011100                  MOV     (R1),R0         ;GET THE NUMBER INPUT
  987  002460  001023                  BNE     90$             ;BR IF NOT DEFAULTED
  988  002462  013700  012130          MOV     DVECT,R0        ;SUBSTITUTE THE DEFAULT
  989  002466  000420                  BR      90$             ;DONE
  990  002470  120127  000005  40$:    CMPB    R1,#5           ;DOES CALLER WANT PRIORITY?
  991  002474  001014                  BNE     80$             ;BR IF INVALID CALL
  992  002476  004537  005372          JSR     R5,PRINT        ;PRINT
  993  002502  013175                  .WORD   PRILV           ;ASK FOR PRIORITY LEVEL
  994  002504  000000                  .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
  995  002506  004537  002532          JSR     R5,INASC        ;REQUEST INPUT
  996  002512  000002                  .WORD   2               ;IN OCTAL
  997  002514  011100                  MOV     (R1),R0         ;GET THE INPUT
  998  002516  001004                  BNE     90$             ;BR IF OK,  NOT DEFAULTED
  999  002520  013700  012132          MOV     DPRIOR,R0       ;SUBSTITUTE THE DEFAULT
 1000  002524  000401                  BR      90$
 1001                                  ;
 1002  002526  000000          80$:    HALT                    ;HALT ON INVALID CALL TO THIS MOD
 1003  002530  000205          90$:    RTS     R5              ;RETURN TO CALLER
 1004                                  .SBTTL  MODULE 1.1.1  INPUT FROM TTY
 1005
```

```
                                ;       MODULE 1.1.1    INPUT FROM TTY
                                ;
                                ;       CALLED AS FOLLOWS
                                ;
                                ;       JSR     R5,INASC
                                ;       .WORD   X
                                ;
                                ;       WHERE   X=1     FOR ASCII INPUT
                                ;               X=2     FOR OCTAL INPUT
                                ;
                                ;       A POINTER TO THE INPUT MESSAGE IS RETURNED IN R1.  THIS
                                ;       ROUTINE DESTROYS R0,R1
                                ;
002532  004737  002620          INASC:  JSR     PC,$TKINT       ;INITIALIZE THE TTY INPUT
002536  012500                          MOV     (R5)+,R0        ;GET THE PASSED PARAMETER
002540  120027  000001                  CMPB    R0,#1           ;IS IT A REQUEST FOR ASCII?
002544  001004                          BNE     10$             ;BR IF NO.
002546  104410                          RDLIN                   ;TRAP TO .$READ MACRO
002550  013637  013632                  MOV     @(SP)+,INBUF    ;GET THE INPUT DATA
002554  000406                          BR      20$             ;GO GET RESULT
                                ;
002556  120027  000002          10$:    CMPB    R0,#2           ;IS IT A REQUEST FOR OCTAL
002562  001006                          BNE     30$             ;BR IF NO
002564  104411                          RDOCT
002566  012637  013632                  MOV     (SP)+,INBUF     ;;POP STACK INTO INBUF
002572  012701  013632          20$:    MOV     #INBUF,R1       ;SET POINTER TO INPUT
002576  000205                          RTS     R5              ;RETURN TO CALLER
002600  000000          30$:    HALT                            ;INVALID CALL TO MODULE
                                ;
                                ;
                                .SBTTL  TTY INPUT ROUTINE
                                ;;*************************************************************************
                                .ENABL  LSB
002602  000000          $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
002604  000000          $TKQIN: .WORD   0               ;;INPUT POINTER
002606  000000          $TKQOUT: .WORD  0               ;;OUTPUT POINTER
002610  000010          $TKQSRT: .BLKB  10              ;;TTY KEYBOARD QUEUE
        002620          $TKQEND=.
                                ;*TK INITIALIZE ROUTINE
                                ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
                                ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
                                ;
                                ;*CALL:
                                ;*      JSR     PC,$TKINT
                                ;*      RETURN
                                ;
002620  005037  002602          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
002624  012737  002610  002604          MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
002632  013737  002604  002606          MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
002640  012737  002670  000060          MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
002646  012737  000200  000062          MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
002654  005777  176266                  TST     @$TKB           ;;CLEAR DONE FLAG
002660  012777  000100  176256          MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
```

```
1062  002666  000207                        RTS     PC              ;;RETURN TO CALLER
1063
1064                                  ;*TK SERVICE ROUTINE
1065                                  ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
1066                                  ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
1067                                  ;*IT IN THE QUEUE.
1068                                  ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
1069                                  ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START)
1070                                  ;
1071  002670  117746  176252          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
1072  002674  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
1073  002700  021627  000021                  CMP     (SP),#$XON      ;;IS IT A RANDOM XON?                   ;RAN001
1074  002704  001002                          BNE     30$             ;;BRANCH IF NO                          ;RAN001
1075  002706  005726                          TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK            ;RAN001
1076  002710  000002                          RTI                     ;;RETURN                                ;RAN001
1077  002712                          30$:                                                                    ;RAN001
1078  002712  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
1079  002716  001007                          BNE     1$              ;;BRANCH IF NO
1080  002720  104401  004022                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
1081  002724  004737  002620                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
1082  002730  005726                          TST     (SP)+           ;;CLEAN UP STACK
1083  002732  000137  001164                  JMP     START           ;;CONTROL C RESTART
1084  002736  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
1085  002742  001004                          BNE     2$              ;;BRANCH IF NO
1086  002744  022737  000176  001140          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
1087  002752  001500                          BEQ     6$              ;;GO TO SWR CHANGE
1088
1089  002754                          2$:
1090  002754  022737  000010  002602          CMP     #10,$TKCNT      ;;IS THE QUEUE FULL?
1091  002762  001004                          BNE     3$              ;;BRANCH IF NO
1092  002764  104401  004016                  TYPE    ,$BELL          ;;RING THE TTY BELL
1093  002770  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
1094  002772  000451                          BR      5$              ;;EXIT
1095  002774  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
1096  003000  001021                          BNE     32$             ;;BRANCH IF NO
1097  003002  005077  176136                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
1098  003006  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
1099  003010  105777  176130          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
1100  003014  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
1101  003016  117746  176124                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
1102  003022  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1103  003026  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
1104  003032  001366                          BNE     31$             ;;BRANCH IF NO
1105  003034  012777  000100  176102          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
1106  003042  000002                          RTI                     ;;RETURN
1107  003044  005237  002602          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
1108  003050  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
1109  003054  002405                          BLT     4$              ;;BRANCH IF YES
1110  003056  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
1111  003062  003002                          BGT     4$              ;;BRANCH IF YES
1112  003064  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
1113  003070  112677  177510          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
1114  003074  005237  002604                  INC     $TKQIN          ;;UPDATE THE POINTER
1115  003100  023727  002604  002620          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
1116  003106  001003                          BNE     5$              ;;BRANCH IF NO
1117  003110  012737  002610  002604          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
```

```
1118  003116  000002                  5$:     RTI                     ;;RETURN
1119
1120                                  ;;**********************************************************************
1121                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1122                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
1123                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
1124                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
1125  003120  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
1126  003126  001124                         BNE     15$             ;;EXIT IF NOT
1127  003130  105777  176010                 TSTB    @$TKS           ;;IS A CHAR WAITING?
1128  003134  100121                         BPL     15$             ;;IF NOT, EXIT
1129  003136  117746  176004                 MOVB    @$TKB,-(SP)     ;;YES
1130  003142  042716  177600                 BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
1131  003146  021627  000007                 CMP     (SP),#7         ;;IS IT A CONTROL-G?
1132  003152  001300                         BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
1133                                                                 ;;AND EXIT
1134
1135                                  ;;**********************************************************************
1136                                  ;*CONTROL IS PASSED TO THIS POINT FROM EIT.  R THE TTY INTERRUPT SERVICE
1137                                  ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
1138                                  ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
1139  003154  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
1140  003162  001674                         BEQ     2$              ;;BRANCH IF YES
1141  003164  005726                         TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
1142  003166  004737  002620                 JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
1143  003172  005077  175746                 CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
1144  003176  112737  000001  001135         MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
1145
1146  003204  104401  004034                 TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
1147  003210  104401  004041  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
1148  003214  013746  000176                 MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
1149  003220  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1150  003222  104401  004052                 TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
1151  003226  005046                 19$:    CLR     -(SP)           ;;CLEAR COUNTER
1152  003230  005046                         CLR     -(SP)           ;;THE NEW SWR
1153  003232  105777  175706         7$:     TSTB    @$TKS           ;;CHAR THERE?
1154  003236  100375                         BPL     7$              ;;IF NOT TRY AGAIN
1155
1156  003240  117746  175702                 MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
1157  003244  042716  177600                 BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1158
1159  003250  021627  000003                 CMP     (SP),#3         ;;IS IT A CONTROL-C?
1160  003254  001015                         BNE     9$              ;;BRANCH IF NOT
1161  003256  104401  004022                 TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
1162  003262  062706  000006                 ADD     #6,SP           ;;CLEAN UP STACK
1163  003266  123727  001135  000001         CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
1164  003274  001003                         BNE     8$              ;;BRANCH IF NO
1165  003276  012777  000100  175640         MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
1166  003304  000137  001164         8$:     JMP     START           ;;CONTROL-C RESTART
1167
1168
1169  003310  021627  000025         9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1170  003314  001005                         BNE     10$             ;;BRANCH IF NOT
1171  003316  104401  004027                 TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
1172  003322  062706  000006         20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1173  003326  000737                         BR      19$             ;;LET'S TRY IT AGAIN
```

```
1174
1175
1176  003330  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
1177  003334  001022                          BNE     16$             ;;BRANCH IF NO
1178  003336  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
1179  003342  001403                          BEQ     11$             ;;BRANCH IF YES
1180  003344  016677  000002  175566          MOV     2(SP),@SWR      ;;SAVE NEW SWR
1181  003352  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
1182  003356  104401  001161          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
1183  003362  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
1184  003370  001003                          BNE     15$             ;;BRANCH IF NOT
1185  003372  012777  000100  175544          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1186  003400  000002                  15$:    RTI                     ;;RETURN
1187  003402  004737  005614          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1188  003406  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
1189  003412  002420                          BLT     18$             ;;BRANCH IF YES
1190  003414  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
1191  003420  003015                          BGT     18$             ;;BRANCH IF YES
1192  003422  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1193  003426  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
1194  003432  001403                          BEQ     17$             ;;BRANCH IF YES
1195  003434  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
1196  003436  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
1197  003440  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
1198  003442  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1199  003446  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1200  003452  000667                          BR      7$              ;;GET THE NEXT ONE
1201  003454  104401  001160          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
1202  003460  000720                          BR      20$             ;;SIMULATE CONTROL-U
1203                                  .DSABL  LSB
1204
1205
1206                                  ;;**************************************************************************
1207                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1208                                  ;*CALL:
1209                                  ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
1210                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1211                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
1212                                  ;
1213
1214  003462  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
1215  003464  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
1216  003472  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
1217  003476  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
1218  003500  012746  003506                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
1219  003504  000002                          RTI                     ;;POP NEW PC AND PS
1220  003506                          64$:
1221  003506  005737  002602          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
1222  003512  001775                          BEQ     1$
1223  003514  005337  002602                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
1224  003520  117766  177062  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
1225  003526  005237  002606                  INC     $TKQOUT         ;;UPDATE THE POINTER
1226  003532  023727  002606  002620          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
1227  003540  001003                          BNE     2$              ;;BRANCH IF NO
1228  003542  012737  002610  002606          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
1229  003550  000002          2$:     RTI                     ;;RETURN
```

```
1230                            ;;**********************************************************************
1231                            ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1232                            ;*CALL:
1233                            ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1234                            ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1235                            ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1236
1237  003552  010346            $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
1238  003554  005046                    CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
1239  003556  012703  004006    1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
1240  003562  022703  004016    2$:     CMP     #$TTYIN+10,R3   ;;BUFFER FULL?
1241  003566  101456                    BLOS    4$              ;;BR IF YES
1242  003570  104407                    RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
1243  003572  112613                    MOVB    (SP)+,(R3)      ;;GET CHARACTER
1244  003574  122713  000177    10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
1245  003600  001022                    BNE     5$              ;;BR IF NO
1246  003602  005716                    TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
1247  003604  001007                    BNE     6$              ;;BR IF NO
1248  003606  112737  000134  004004    MOVB    #'\,9$          ;;TYPE A BACK SLASH
1249  003614  104401  004004            TYPE    ,9$
1250  003620  012716  177777            MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
1251  003624  005303            6$:     DEC     R3              ;;BACKUP BY ONE
1252  003626  020327  004006            CMP     R3,#$TTYIN      ;;STACK EMPTY?
1253  003632  103434                    BLO     4$              ;;BR IF YES
1254  003634  111337  004004            MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
1255  003640  104401  004004            TYPE    ,9$             ;;GO TYPE
1256  003644  000746                    BR      2$              ;;GO READ ANOTHER CHAR.
1257  003646  005716            5$:     TST     (SP)            ;;RUBOUT KEY SET?
1258  003650  001406                    BEQ     7$              ;;BR IF NO
1259  003652  112737  000134  004004    MOVB    #'\,9$          ;;TYPE A BACK SLASH
1260  003660  104401  004004            TYPE    ,9$
1261  003664  005016                    CLR     (SP)            ;;CLEAR THE RUBOUT KEY
1262  003666  122713  000025    7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
1263  003672  001003                    BNE     8$              ;;BR IF NO
1264  003674  104401  004027            TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
1265  003700  000726                    BR      1$              ;;GO START OVER
1266  003702  122713  000022    8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
1267  003706  001011                    BNE     3$              ;;BRANCH IF NO
1268  003710  105013                    CLRB    (R3)            ;;CLEAR THE CHARACTER
1269  003712  104401  001161            TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
1270  003716  104401  004006            TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
1271  003722  000717                    BR      2$              ;;GO PICKUP ANOTHER CHACTER
1272  003724  104401  001160    4$:     TYPE    ,$QUES          ;;TYPE A '?'
1273  003730  000712                    BR      1$              ;;CLEAR THE BUFFER AND LOOP
1274  003732  111337  004004    3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
1275  003736  104401  004004            TYPE    ,9$
1276  003742  122723  000015            CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1277  003746  001305                    BNE     2$              ;;LOOP IF NOT RETURN
1278  003750  105063  177777            CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1279  003754  104401  001162            TYPE    ,$LF            ;;TYPE A LINE FEED
1280  003760  005726                    TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
1281  003762  012603                    MOV     (SP)+,R3        ;;RESTORE R3
1282  003764  011646                    MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1283  003766  016666  000004  000002    MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1284  003774  012766  004006  000004    MOV     #$TTYIN,4(SP)
1285  004002  000002                    RTI                     ;;RETURN
```

```
                                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
                                        .BYTE   0               ;;TERMINATOR
                                $TTYIN: .BLKB   10              ;;RESERVE 10 BYTES FOR TTY INPUT
                                $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
                                $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
                                $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
                                $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
                                $MSWR:  .ASCIZ  <15><12>/SWR = /

                                $MNEW:  .ASCIZ  /  NEW = /

                                .EVEN
                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                ;;*******************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT

004064  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
004066  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
004074  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
004076  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
004100  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
004102  104410                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
004104  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
004106  010037  004212                  MOV     R0,5$           ;;AND SAVE IT
004112  005001                          CLR     R1              ;;CLEAR DATA WORD
004114  005002                          CLR     R2
004116  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
004120  001420                          BEQ     3$              ;;IF ZERO GET OUT
004122  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
004126  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
004130  122716  000067                  CMPB    #'7,(SP)
004134  002423                          BLT     4$
004136  006301                          ASL     R1              ;;*2
004140  006102                          ROL     R2
004142  006301                          ASL     R1              ;;*4
004144  006102                          ROL     R2
004146  006301                          ASL     R1              ;;*8
004150  006102                          ROL     R2
004152  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
004156  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
004160  000756                          BR      2$              ;;LOOP
004162  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
004164  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
004170  010237  004222                  MOV     R2,$HIOCT
004174  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
004176  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
1342   004200  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1343   004202  000002                          RTI                     ;;RETURN
1344   004204  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
1345   004206  105010                          CLRB    (R0)            ;;SET A TERMINATOR
1346   004210  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
1347   004212  000000                  5$:     .WORD   0
1348   004214  104401  001160                  TYPE    ,$QUES          ;;"?" "CR" & "LF"
1349   004220  000730                          BR      1$              ;;TRY AGAIN
1350   004222  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
1351                                           .SBTTL  MODULE 1.2  SIZE MEMORY
1352
```

```
1353                                    ;       MODULE 1.2      SIZE MEMORY
1354                                    ;
1355                                    ;       CALLED AS FOLLOWS
1356                                    ;
1357                                    ;       JSR     PC,SIZE
1358                                    ;
1359                                    ;       THIS MODULE SETS UP MEMORY SIZE INCLUDING EXTENDED
1360                                    ;       ADDRESSING BITS IN ACTIVE PARAMETER TABLE
1361                                    ;
1362                                    ;
1363                                    ;
1364  004224  005037  004574           SIZE:   CLR     $KT11           ;DON'T TRY FOR MEM MGT
1365  004230  004737  004510                   JSR     PC,$SIZE        ;GET MEM SIZE WITH KT OFF
1366  004234  005037  012172                   CLR     KT              ;RESET KT AVAILABLE SWITCH
1367  004240  013737  005220  012166           MOV     $LSTAD,MMAX     ;SAVE END OF CORE WITH KT OFF
1368  004246  013737  005220  012164           MOV     $LSTAD,MSIZE    ;GET END OF CORE TO WORK WITH
1369  004254  023727  005220  017776           CMP     $LSTAD,#17776   ;LESS THAN 8K?
1370  004262  103415                           BLO     35$             ;BRANCH IF YES, DON'T PRESERVE LOADERS
1371  004264  162737  000276  012164           SUB     #276,MSIZE      ;SAVE SPACE FOR ABS LOADER
1372  004272  005737  000042                   TST     @#42            ;RUNNING UNDER ACT OR XXDP
1373  004276  001407                           BEQ     35$             ;BR IF NO
1374  004300  023737  000042  000046           CMP     @#42,@#46       ;RUNNING UNDER XXDP CHAIN?
1375  004306  001403                           BEQ     35$             ;BR IF NO
1376  004310  162737  005500  012164           SUB     #5500,MSIZE     ;SAVE SPACE FOR LOADERS
1377  004316  005037  012170           35$:    CLR     EASIZE          ;CLEAN UP EXTENDED ADDR
1378  004322  005037  012224                   CLR     EABITS
1379  004326  037727  174606  002000           BIT     @SWR,#SW10      ;CHECK IF MM TO BE USED
1380  004334  001064                           BNE     30$             ;BR IF NO
1381  004336  012737  000200  004574   10$:    MOV     #200,$KT11
1382  004344  004737  004510           20$:    JSR     PC,$SIZE
1383  004350  005737  004574                   TST     $KT11           ;IS MM ACTIVE
1384  004354  100054                           BPL     30$             ;BR IF NO
1385  004356  012737  000001  012172           MOV     #1,KT           ;SET KT IN USE FLAG
1386  004364  022737  007540  005222           CMP     #7540,$LSTBK    ;;124KW ?               VRG
1387  004372  003003                           BGT     1$              ;;NO - BRANCH           VRG
1388  004374  012737  007540  005222           MOV     #7540,$LSTBK    ;;FORCE IT TO 124KW     VRG
1389  004402  013737  005222  012170   1$:     MOV     $LSTBK,EASIZE
1390  004410  062737  000040  012170           ADD     #40,EASIZE      ;INCR TO FIRST INVALID SETTING
1391  004416  013737  005222  012226           MOV     $LSTBK,PHYMAX   ;START GEN OF MAX PHYSICAL
1392  004424  006337  012226                   ASL     PHYMAX          ;  ADDR
1393  004430  006337  012226                   ASL     PHYMAX          ;    LOW
1394  004434  006337  012226                   ASL     PHYMAX          ;      16
1395  004440  006337  012226                   ASL     PHYMAX          ;        BITS
1396  004444  006337  012226                   ASL     PHYMAX
1397  004450  006337  012226                   ASL     PHYMAX
1398  004454  052737  003776  012226           BIS     #3776,PHYMAX    ;SET UP TO USE LAST 1K TOO
1399  004462  013737  005222  012224           MOV     $LSTBK,EABITS
1400  004470  042737  171777  012224           BIC     #171777,EABITS  ;SAVE ONLY EXT ADDR BITS
1401  004476  000337  012224                   SWAB    EABITS          ;MOVE TO BIT POS. 3,2
1402  004502  006337  012224                   ASL     EABITS          ;MOVE TO BIT POS. 4,3 FOR DX11
1403  004506  000207                   30$:    RTS     PC              ;RETURN
1404                                    ;
1405                                    .SBTTL  ROUTINE TO SIZE MEMORY
1406
1407                                    ;;**********************************************************************
1408                                    ;*CALL:
```

```
                                ;*      JSR     PC,$SIZE
                                ;*      RETURN
                                ;*$LSTAD WILL CONTAIN:
                                ;*      WITH KT11 OPTION        -- LAST VIRTUAL ADDRESS OF THE LAST BANK
                                ;*      WITHOUT KT11 OPTION     -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
                                ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
                                ;*$KT11 IS THE MEMORY MANAGEMENT KEY
                                ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
                                ;*      MUST BE SETUP BEFORE THE CALL
                                ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
                                ;*      DETERMINED BY ROUTINE

004510  010046          $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
004512  010146                  MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
004514  010246                  MOV     R2,-(SP)        ;;SAVE R2 ON THE STACK
004516  010346                  MOV     R3,-(SP)        ;;SAVE R3 ON THE STACK
004520  010446                  MOV     R4,-(SP)        ;;SAVE R4 ON THE STACK                          BK001
004522  013746  000114          MOV     @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
004526  013746  000116          MOV     @#116,-(SP)
004532  012737  000116  000114  MOV     #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
004540  012737  000002  000116  MOV     #RTI,@#116
004546  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
004552  013746  000006          MOV     @#ERRVEC+2,-(SP)
004556  010600                  MOV     SP,R0           ;;SAVE THE STACK POINTER
                        ;;SET THE ERRVEC PS TO THE PRESENT PS
004560  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
004562  012637  000006          MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
004566  012701  003776          MOV     #3776,R1        ;;SETUP ADDRESS
004572  105727                  TSTB    (PC)+           ;;USE MEMORY MANAGEMENT?
004574  000200          $KT11:  .WORD   200             ;;SET TO USE MEMORY MANAGEMENT
004576  100145                  BPL     $CORE           ;;BR IF NO
004600  012737  005104  000004  MOV     #$KTNEX,@#ERRVEC ;;SET FOR TIMEOUT
004606  005737  177572          TST     @#SR0           ;;KT11 ARE YOU THERE?
004612  052737  100000  004574  BIS     #100000,$KT11   ;;YES--SET KT11 KEY
004620  012737  004650  000004  MOV     #100$,@#ERRVEC  ;;SET FOR TIMEOUT                               BK001
004626  005737  170200          TST     @#170200        ;;UNIBUS MAP ARE YOU THERE?                     BK001
004632  012737  176200  004670  MOV     #176200,@#$STOP ;;YES-SET COMPARISON VALUE FOR 11/70            BK001
004640  012737  000200  004666  MOV     #200,@#$MAP     ;;TURN ON MAP INDICATOR                         BK001
004646  000411                  BR      $MAPRG          ;;GO SET UP MAP REGISTERS                       BK001
004650  012737  006200  004670  100$:   MOV     #6200,@#$STOP   ;;COMPARISON VALUE FOR 18 BIT MAPPING   BK001
004656  022626                  CMP     (SP)+,(SP)+     ;;CLEAN OFF STACK                               BK001
004660  005037  004666          CLR     @#$MAP          ;;MAKE SURE MAP INDICATOR TURNED OFF            BK001
004664  000412                  BR      $NOMAP          ;;                                      BK001
004666  000000          $MAP:   .WORD   0               ;;=200 IF MAP PRESENT                           BK001
004670  000000          $STOP:  .WORD   0               ;;FILLED WITH APPROPRIATE COMPARISON VALUE              BK001
004672  012703  000037  $MAPRG: MOV     #37,R3          ;;SET UP COUNTER                                BK001
004676  012702  170200          MOV     #170200,R2      ;;START WITH MAPLO                              BK001
004702  005022          100$:   CLR     (R2)+           ;;LOAD ALL MAP REGISTERS                        BK001
004704  012722  000074          MOV     #74,(R2)+       ;;WITH THE VALUE 17000000                       BK001
004710  077304                  SOB     R3,100$         ;;DO ALL 31 REGISTERS                           BK001
004712          $NOMAP:
004712  005046                  CLR     -(SP)           ;;INITIALIZE FOR "PAR" LOADING
004714  012702  172340          MOV     #KIPAR0,R2      ;;ADDRESS OF FIRST "PAR"
004720  012703  000010          MOV     #^D8,R3         ;;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
004724  012762  077406  177740  1$:     MOV     #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
004732  011622                  MOV     (SP),(R2)+      ;;LOAD "PAR"
```

```
1465  004734  062716  000200                ADD     #200,(SP)       ;;UPDATE FOR NEXT "PAR"
1466  004740  077307                        SOB     R3,1$           ;;LOOP UNTIL ALL EIGHT ARE LOADED
1467  004742  012742  177600                MOV     #177600,-(R2)   ;;SETUP KIPAR7 FOR I/O
1468  004746  005042                        CLR     -(R2)           ;;SETUP KIPAR6 FOR TESTING
1469  004750  012737  004766  000004        MOV     #2$,@#ERRVEC    ;;CATCH TIMEOUT IF NO SR3
1470  004756  012737  000060  172516        MOV     #60,@#SR3       ;;ENABLE 22 BIT MODE AND UNIBUS MAP     BK001
1471  004764  000401                        BR      3$              ;;THIS PDP-11 HAS A SR3 REGISTER
1472  004766  022626                2$:     CMP     (SP)+,(SP)+     ;;CLEAN OFF THE STACK--NO SR3
1473  004770  005237  177572        3$:     INC     @#SR0           ;;TURN ON MEMORY MANAGEMENT
1474  004774  012737  005042  000004        MOV     #$KTOUT,@#ERRVEC ;;SET FOR TIME OUT
1475  005002  105737  004666                TSTB    @#$MAP          ;;IS MAP THERE?                         BK001
1476  005006  100006                        BPL     4$              ;;NO-SKIP                               BK001
1477  005010  012737  005064  000114        MOV     #$MMOUT,@#114   ;;SET UP MEMORY ERROR VECTOR            BK001
1478  005016  013737  000006  000116        MOV     @#ERRVEC+2,@#116 ;;LOCK OUT INTERRUPTS          BK001
1479  005024  005737  143776        4$:     TST     @#143776        ;;TRAP ON NON-EX-MEM
1480  005030  062712  000040                ADD     #40,(R2)        ;;MAKE A 1K STEP
1481  005034  023712  004670                CMP     @#$STOP,(R2)    ;;LAST ONE?
1482  005040  101371                        BHI     4$              ;;NO--TRY IT
1483  005042  011202        $KTOUT: MOV     (R2),R2         ;;GET LAST BANK+1
1484  005044  005037  177572                CLR     @#SR0           ;;TURN OFF MEMORY MANAGEMENT
1485  005050  105737  004666                TSTB    @#$MAP          ;;IS MAP THERE?                         BK001
1486  005054  100034                        BPL     $SIZEX          ;;NO-SKIP                               BK001
1487  005056  005037  172516                CLR     @#SR3           ;;TURN OFF MAP                          BK001
1488  005062  000431                        BR      $SIZEX
1489  005064  013704  177744        $MMOUT: MOV     @#177744,R4     ;;SAVE MEMORY ERROR REGISTER            BK001
1490  005070  010437  177744                MOV     R4,@#177744     ;;CLEAR BITS IN REGISTER                BK001
1491  005074  032704  000001                BIT     #1,R4           ;;MEMORY TIMEOUT?                       BK001
1492  005100  001360                        BNE     $KTOUT          ;;YES-EXIT                              BK001
1493  005102  000002                        RTI                     ;;MUST BE PARITY ERROR-IGNORE IT        BK001
1494  005104  042737  100000  004574 $KTNEX: BIC     #100000,$KT11   ;;KT11 NON-EXISTENT
1495  005112  012737  005142  000004 $CORE:  MOV     #$CROUT,@#ERRVEC ;;SET FOR TIMEOUT
1496  005120  005002                        CLR     R2              ;;SET UP BANK
1497  005122  062701  004000        1$:     ADD     #4000,R1        ;;INCREMENT BY 1K
1498  005126  062702  000040                ADD     #40,R2          ;;1K STEP
1499  005132  005711                        TST     (R1)            ;;TRAP ON TIME OUT
1500  005134  022701  177776                CMP     #177776,R1      ;;LAST ONE
1501  005140  001370                        BNE     1$              ;;NO--TRY AGAIN
1502  005142  162701  004000        $CROUT: SUB     #4000,R1
1503  005146  162702  000040        $SIZEX: SUB     #40,R2          ;;DROP BACK
1504  005152  010006                        MOV     R0,SP           ;;RESTORE THE STACK
1505  005154  012637  000006                MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
1506  005160  012637  000004                MOV     (SP)+,@#ERRVEC
1507  005164  012637  000116                MOV     (SP)+,@#116     ;;RESTORE MEMORY ERROR VECTOR
1508  005170  012637  000114                MOV     (SP)+,@#114
1509  005174  010137  005220                MOV     R1,$LSTAD       ;;LAST ADDRESS
1510  005200  010237  005222                MOV     R2,$LSTBK       ;;LAST BANK
1511  005204  012604                        MOV     (SP)+,R4        ;;RESTORE R4                            BK001
1512  005206  012603                        MOV     (SP)+,R3        ;;RESTORE R3
1513  005210  012602                        MOV     (SP)+,R2        ;;RESTORE R2
1514  005212  012601                        MOV     (SP)+,R1        ;;RESTORE R1
1515  005214  012600                        MOV     (SP)+,R0        ;;RESTORE R0
1516  005216  000207                        RTS     PC
1517  005220  000000        $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
1518  005222  000000        $LSTBK: .WORD   0               ;;CONTAINS THE LAST BANK
1519
1520                                  .SBTTL  MODULE 1.3  CHECK PARAMETERS
```

```
1521                                    ;       MODULE 1.3      CHECK PARAMETERS
1522                                    ;
1523                                    ;       CALLED AS FOLLOWS
1524                                    ;
1525                                    ;       JSR     R5,CHECKP
1526                                    ;       .WORD   X
1527                                    ;       THIS MODULE HAS RESPONSIBILITY FOR VALIDATING PARAMETERS
1528                                    ;       INPUT BY THE OPERATOR.  IT COMPARES THE INPUT PARAMETER
1529                                    ;       AGAINST PARAMETER LIMITS DEFINED IN THE INPUT PARAMETER
1530                                    ;       LIMITS TABLE.  IF THE PARAMETER IS OUTSIDE THE SPECIFIED
1531                                    ;       LIMITS, AN ERROR MESSAGE IS OUTPUT AND ALL ONES IS RETURNED
1532                                    ;       IN R0.  IF THE PARAMETER IS OK, ALL ZEROS IS RETURNED IN
1533                                    ;       R0.
1534                                    ;
1535                                    ;       X DEFINES THE PARAMETER TO CHECK AS FOLLOWS:
1536                                    ;       X=2     STARTING TEST NUMBER
1537                                    ;       X=3     BASE ADDR OF DX CONTROL REGS
1538                                    ;       X=4     DX VECTOR ADDRESS
1539                                    ;       X=5     DX PRIORITY.
1540                                    ;       THE PARAMETER TO BE CHECKED IS PASSED IN R0.
1541                                    ;
1542  005224  012501                    CHECKP: MOV     (R5)+,R1        ;GET PARAMETER TO BE CHECKED
1543  005226  120127  000002                    CMPB    R1,#2           ;IS IT TEST NO?
1544  005232  001007                            BNE     10$             ;BR IF NO
1545  005234  020037  012134                    CMP     R0,LTESTN       ;CHECK LOW LIMIT
1546  005240  002443                            BLT     90$             ;BR IF ERROR
1547  005242  020037  012136                    CMP     R0,HTESTN       ;CHECK HIGH LIMIT
1548  005246  003040                            BGT     90$             ;BR IF ERROR
1549  005250  000446                            BR      80$             ;GO TO OK RETURN
1550  005252  120127  000003           10$:     CMPB    R1,#3           ;IS IT BASE ADDR OF DX
1551  005256  001007                            BNE     20$             ;BR IF NO
1552  005260  020037  012140                    CMP     R0,LUBA         ;CHECK LOW LIMIT
1553  005264  002431                            BLT     90$             ;BR IF BAD
1554  005266  020037  012142                    CMP     R0,HUBA         ;CHECK HIGH LIMIT
1555  005272  003026                            BGT     90$             ;BR IF BAD
1556  005274  000434                            BR      80$             ;OK, RETURN
1557  005276  120127  000004           20$:     CMPB    R1,#4           ;IS IT THE VECTOR?
1558  005302  001007                            BNE     30$             ;BR IF NO
1559  005304  020037  012144                    CMP     R0,LVEC         ;CHECK THE LOW LIMIT
1560  005310  002417                            BLT     90$             ;BR IF BAD
1561  005312  020037  012146                    CMP     R0,HVEC         ;CHECK THE HIGH LIMIT
1562  005316  003014                            BGT     90$             ;BR IF BAD
1563  005320  000422                            BR      80$             ;OK, RETURN
1564  005322  120127  000005           30$:     CMPB    R1,#5           ;IS PARM PRIORITY LEVEL
1565  005326  001007                            BNE     40$             ;BR IF NO
1566  005330  020037  012150                    CMP     R0,LPRIOR       ;CHECK LOW LIMIT
1567  005334  002405                            BLT     90$             ;BR IF BAD
1568  005336  020037  012152                    CMP     R0,HPRIOR       ;CHECK HIGH LIMIT
1569  005342  003002                            BGT     90$             ;BR IF BAD
1570  005344  000410                            BR      80$             ;OK, RETURN
1571  005346  000000           40$:     HALT                            ;ERROR, BAD MODULE CALL PARM
1572  005350  004537  005372           90$:     JSR     R5,PRINT        ;PRINT PARM LIMIT ERROR
1573  005354  013250                            .WORD   BADP            ;ADDR OF MESSAGE
1574  005356  000000                            .WORD   0               ;CONTROL WORD, CRLF + DON'T CONVERT
1575  005360  012700  177777                    MOV     #-1,R0          ;SET ERROR RETURN CODE
1576  005364  000401                            BR      95$             ;RETURN
```

```
  1577  005366  005000                  80$:    CLR     R0              ;SET GOOD RETURN CODE
  1578  005370  000205                  95$:    RTS     R5              ;RETURN TO CALLER.
  1579
  1580                                          .SBTTL  MODULE 1.3.1  WRITE TTY
```

```
                                ;       MODULE 1.3.1    WRITE TTY
                                ;
                                ;       THIS MODULE HANDLES ALL OUTPUT TO THE OPERATOR
                                ;
                                ;       CALLED AS FOLLOWS
                                ;
                                ;       JSR     R5      PRINT
                                ;       .WORD   MSG             ;ADDRESS OF MESSAGE OR OCT #
                                ;       .WORD   CTL
                                ;       RETURN
                                ;
                                ;       CTL=    BIT0=1          ;DONT - CRLF BEFORE MESSAGE
                                ;               BIT15=1         ;CONVERT OCTAL TO ASCII AND PRINT
                                ;
005372                          PRINT:
005372  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
005374  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
005376  012500                         MOV     (R5)+,R0        ;GET ADDR OF MSG
005400  012501                         MOV     (R5)+,R1        ;GET PRINT CONTROL WORD
005402  032701  000001                 BIT     #1,R1           ;DO WE NEED A CRLF?
005406  001002                         BNE     10$             ;BR IF NO
005410  104401                         TYPE                    ;TRAP TO TYPE ROUTINE
005412  013625                         .WORD   #CRLF1          ;POINTER TO CRLF ASCII
005414  005701                 10$:    TST     R1              ;IS IT OCTAL?
005416  100003                         BPL     20$             ;BR IF NO
005420  011046                         MOV     (R0),-(SP)      ;PUT OCTAL # ON STACK
005422  104402                         TYPOC                   ;TYPE 6 CHAR VIA TRAP
005424  000404                         BR      40$             ;DONE
005426  010037  005434         20$:    MOV     R0,25$          ;PUT MESSAGE ADDR INLINE
005432  104401                         TYPE                    ;TRAP TO TYPE ASCII RTN
005434  000000                 25$:    .WORD   0               ;POINTER TO MSG GOES HERE
005436                          40$:
005436  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
005440  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
005442  000205                         RTS     R5              ;RETURN TO CALLER
                                ;TYPE TRAP CALL COMES HERE
                                .SBTTL  TYPE ROUTINE

                                ;;**********************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*

005444  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
005450  100002                         BPL     1$              ;;BR IF YES
```

```
1637  005452  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
1638  005454  000407                        BR      3$              ;;LEAVE
1639  005456  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
1640  005460  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1641  005464  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1642  005466  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1643  005470  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1644  005472  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
1645  005474  062716  000002        3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1646  005500  000002                        RTI                     ;;RETURN
1647  005502  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1648  005506  001430                        BEQ     8$
1649  005510  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1650  005514  001006                        BNE     5$
1651  005516  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
1652  005520  104401                        TYPE                    ;;TYPE A CR AND LF
1653  005522  001161                        $CRLF
1654  005524  105037  005742                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
1655  005530  000755                        BR      2$              ;;GET NEXT CHARACTER
1656  005532  004737  005614        5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1657  005536  123726  001156        6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1658  005542  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
1659  005544  013746  001154                MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1660                                                                ;;AND THE NULL CHAR.
1661  005550  105366  000001        7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1662  005554  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1663  005556  004737  005614                JSR     PC,$TYPEC       ;;GO TYPE A NULL
1664  005562  105337  005742                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1665  005566  000770                        BR      7$              ;;LOOP
1666
1667                                ;HORIZONTAL TAB PROCESSOR
1668
1669  005570  112716  000040        8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1670  005574  004737  005614        9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1671  005600  132737  000007  005742        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1672  005606  001372                        BNE     9$              ;;TAB STOP
1673  005610  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
1674  005612  000724                        BR      2$              ;;GET NEXT CHARACTER
1675  005614                        $TYPEC:
1676  005614  105777  173324                TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
1677  005620  100022                        BPL     10$             ;;BR IF NOT                     ;MJD001
1678  005622  017746  173320                MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
1679  005626  042716  177600                BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
1680  005632  122716  000023                CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
1681  005636  001012                        BNE     102$            ;;BR IF NOT                     ;MJD001
1682  005640                        101$:                           ;                               ;MJD001
1683  005640  105777  173300                TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
1684  005644  100375                        BPL     101$            ;                               ;MJD001
1685  005646  117716  173274                MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
1686  005652  042716  177600                BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
1687  005656  122716  000021                CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
1688  005662  001366                        BNE     101$            ;;BR IF NOT                     ;MJD001
1689  005664                        102$:                           ;                               ;MJD001
1690  005664  005726                        TST     (SP)+           ;;FIX STACK                     ;MJD001
1691  005666                        10$:                            ;                               ;MJD001
1692  005666  105777  173256                TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
```

```
1693  005672  100375                    BPL     10$                                                     ;MJD001
1694  005674  126627  000002  000021    CMPB    2(SP),#$XON     ;;IS CHARACTER A RANDOM XON?            ;RAN001
1695  005702  001420                    BEQ     $TYPEX          ;;BRANCH IF YES                         ;RAN001
1696  005704  116677  000002  173240    MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1697  005712  122766  000015  000002    CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1698  005720  001003                    BNE     1$              ;;BRANCH IF NO
1699  005722  105037  005742            CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1700  005726  000406                    BR      $TYPEX          ;;EXIT
1701  005730  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1702  005736  001402                    BEQ     $TYPEX          ;;BRANCH IF YES
1703  005740  105227                    INCB    (PC)+           ;;COUNT THE CHARACTER
1704  005742  000000                  $CHARCNT:.WORD  0         ;;CHARACTER COUNT STORAGE
1705  005744  000207                  $TYPEX: RTS     PC
1706
1707                                  ;TYPOC TRAP CALL COMES TO $TYPOC TAG BELOW
1708                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1709
1710                                  ;;**********************************************************************
1711                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1712                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1713                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1714                                  ;*CALL:
1715                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1716                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
1717                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1718                                  ;*      .BYTE   M               ;;M=1 OR 0
1719                                  ;*                                      ;;1=TYPE LEADING ZEROS
1720                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
1721                                  ;*
1722                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1723                                  ;*$TYPOS OR $TYPOC
1724                                  ;*CALL:
1725                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1726                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
1727                                  ;*
1728                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1729                                  ;*CALL:
1730                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1731                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
1732
1733  005746  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
1734  005752  116637  000001  006171          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
1735  005760  112637  006173                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
1736  005764  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
1737  005770  000406                          BR      $TYPON
1738  005772  112737  000001  006171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
1739  006000  112737  000006  006173          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
1740  006006  112737  000005  006170  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
1741  006014  010346                          MOV     R3,-(SP)        ;;SAVE R3
1742  006016  010446                          MOV     R4,-(SP)        ;;SAVE R4
1743  006020  010546                          MOV     R5,-(SP)        ;;SAVE R5
1744  006022  113704  006173                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
1745  006026  005404                          NEG     R4
1746  006030  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
1747  006034  110437  006172                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
1748  006040  113704  006171                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
```

```
1749  006044  016605  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
1750  006050  005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
1751  006052  006105              1$: ROL     R5              ;;ROTATE MSB INTO "C"
1752  006054  000404                  BR      3$              ;;GO DO MSB
1753  006056  006105              2$: ROL     R5              ;;FORM THIS DIGIT
1754  006060  006105                  ROL     R5
1755  006062  006105                  ROL     R5
1756  006064  010503                  MOV     R5,R3
1757  006066  006103              3$: ROL     R3              ;;GET LSB OF THIS DIGIT
1758  006070  105337  006172          DECB    $OMODE          ;;TYPE THIS DIGIT?
1759  006074  100016                  BPL     7$              ;;BR IF NO
1760  006076  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
1761  006102  001002                  BNE     4$              ;;TEST FOR 0
1762  006104  005704                  TST     R4              ;;SUPPRESS THIS 0?
1763  006106  001403                  BEQ     5$              ;;BR IF YES
1764  006110  005204              4$: INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
1765  006112  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
1766  006116  052703  000040      5$: BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
1767  006122  110337  006166          MOVB    R3,8$           ;;SAVE FOR TYPING
1768  006126  104401  006166          TYPE    ,8$             ;;GO TYPE THIS DIGIT
1769  006132  105337  006170      7$: DECB    $OCNT           ;;COUNT BY 1
1770  006136  003347                  BGT     2$              ;;BR IF MORE TO DO
1771  006140  002402                  BLT     6$              ;;BR IF DONE
1772  006142  005204                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
1773  006144  000744                  BR      2$              ;;GO DO THE LAST DIGIT
1774  006146  012605              6$: MOV     (SP)+,R5        ;;RESTORE R5
1775  006150  012604                  MOV     (SP)+,R4        ;;RESTORE R4
1776  006152  012603                  MOV     (SP)+,R3        ;;RESTORE R3
1777  006154  016666  000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
1778  006162  012616                  MOV     (SP)+,(SP)
1779  006164  000002                  RTI                     ;;RETURN
1780  006166     000              8$: .BYTE   0               ;;STORAGE FOR ASCII DIGIT
1781  006167     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
1782  006170     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
1783  006171     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
1784  006172  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1785                                  .SBTTL  MODULE 2.0  INTERRUPT HANDLERS
```

```
                                        ;
                                        ;       MODULE 2.0
                                        ;
                                        ;
                                        ;       INTERRUPT HANDLERS
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        .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.

006174  016646  000002         $TRAP:  MOV     2(SP),-(SP)     ;;ASSUME THE STATUS OF
006200  042716  000020                 BIC     #20,(SP)        ;; THE CALLER--DO NOT ALLOW
006204  012746  006212                 MOV     #1$,-(SP)       ;; T-BIT TRAPS
006210  000002                         RTI                     ;;SET THE NEW STATUS
006212  010046                 1$:     MOV     R0,-(SP)        ;;SAVE R0
006214  016600  000002                 MOV     2(SP),R0        ;;GET TRAP ADDRESS
006220  005740                         TST     -(R0)           ;;BACKUP BY 2
006222  111000                         MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
006224  006300                         ASL     R0              ;;POSITION FOR INDEXING
006226  016000  006246                 MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
006232  000200                         RTS     R0              ;;GO TO ROUTINE


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

006234  011646                 $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
006236  016666  000004  000002         MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
006244  000002                         RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
006246  006234                 $TRPAD: .WORD   $TRAP2
006250  005444                         $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
006252  005772                         $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
006254  005746                         $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
006256  006006                         $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

006260  003210                         $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

006262  003120                         $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
006264  003462                         $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
006266  003552                         $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
006270  004064                         $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
```

```
1842
1843                                    .SBTTL  POWER DOWN AND UP ROUTINES
1844
1845                                    ;;**********************************************************************
1846                                    ;POWER DOWN ROUTINE
1847   006272  012737  006436  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
1848   006300  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
1849   006306  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1850   006310  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1851   006312  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1852   006314  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1853   006316  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1854   006320  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1855   006322  017746  172612                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
1856   006326  010637  006442                  MOV     SP,$SAVR6       ;;SAVE SP
1857   006332  012737  006344  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
1858   006340  000000                          HALT
1859   006342  000776                          BR      .-2             ;;HANG UP
1860
1861                                    ;;**********************************************************************
1862                                    ;POWER UP ROUTINE
1863   006344  012737  006436  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
1864   006352  013706  006442                  MOV     $SAVR6,SP       ;;GET SP
1865   006356  005037  006442                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
1866   006362  005237  006442          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
1867   006366  001375                          BNE     1$              ;;OF  WORD
1868   006370  012677  172544                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
1869   006374  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1870   006376  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
1871   006400  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1872   006402  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1873   006404  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1874   006406  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1875   006410  012737  006272  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
1876   006416  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
1877   006424  104401                          TYPE                    ;REPORT THE POWER FAILURE
1878   006426  006444          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
1879   006430  012716                  MOV     (PC)+,(SP)              ;;RESTART AT RESTART
1880   006432  001334          $PWRAD: .WORD   RESTART                 ;;RESTART ADDRESS
1881   006434  000002                  RTI
1882   006436  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
1883   006440  000776                  BR      .-2                     ;; BEFORE THE POWER DOWN WAS COMPLETE
1884   006442  000000          $SAVR6: 0                               ;;PUT THE SP HERE
1885   006444  005015  047520  042527  $POWER: .ASCIZ   <15><12>''POWER''
1886   006452  000122
1887                                            .EVEN
```

```
                        .SBTTL  MODULE 3.0.1  TST1  FAST NPR DATA TEST
                ;
                ;       MODULE 3.0.1  FAST NPR DATA TEST
                ;
                ;       THIS IS A FUNCTIONAL TEST TO CHECK THE ABILITY
                ;       OF THE DX TO DO NPR'S OF DATA INTO MEMORY
                ;       WITH LARGE BYTE COUNTS.  THE DXBA IS CHECKED FOR
                ;       ITS ABILITY TO COUNT AND ALL CARRIES IN THE COUNTER ARE
                ;       EXERCISED.  THE DATA TRANSFERS ACCESS ALL
                ;       AVAILABLE MEMORY EXCEPT WHERE THE PROGRAM IS LOCATED,
                ;       AND THE AREA WHICH MAY BE OCCUPIED BY A LOADER.
                ;
006454 004537 007050    TST1:   JSR     R5,INIT         ;INITIALIZE THE SOFTWARE, PS, ETC
006460 004737 007100            JSR     PC,DXRES        ;PUT THE DX IN A GOOD KNOWN STATE
006464 004537 010620            JSR     R5,BUFAD        ;GET FIRST BUFFER BASE AND SET UP KT
006470 100402                   .WORD   100402          ;CTL BIT 15=INITIALIZE, BUFFER BASE DIVISABLE
                                                        ;BY 400, SET UP BUFMAX
006472 013700 012160            MOV     AVECT,R0        ;SET UP
006476 012720 006550            MOV     #INTR1,(R0)+    ;DX INTERRUPT
006502 012710 000340            MOV     #340,(R0)       ;AND DX PSW
006506 012737 000402 012232 LOOP1:  MOV     #402,BUFSZE     ;BUFFER SIZE A LITTLE BIGGER THAN OFFSET
                                                        ;TO CREATE A CARRY OUT OF HIGH ORDER BIT
006514 004737 007030        10$:    JSR     PC,BUFCLR       ;CLEAR THE BUFFER
006520 012737 100525 012120         MOV     #100525,DXM00   ;SET DATA PATTERN AND OPOUT
006526 053777 012236 003324 GO1:    BIS     EACUR,@DXCSA    ;SET UP DX EXT ADDR BITS
006534 013777 012242 003322         MOV     BFBAPH,@DXBAA   ;SET UP DX BUS ADDR REG.
006542 004537 006706                JSR     R5,DXGO         ;START THE DX
006546 177376                       .WORD   -402            ;BYTE COUNT

                ;INTERRUPT WILL RETURN HERE AFTER DATA TRANSFER IS COMPLETE

006550 005737 007026    INTR1:  TST     INTERR          ;DID WE GET HERE VIA INTER
006554 001006                   BNE     8$              ;BR IF NO, STACK IS OK
006556 012716 006564            MOV     #5$,@SP         ;SET UP RETURN ADDR
006562 000002                   RTI                     ;RELIEVE INTERRUPT STATUS
006564 012705 006572    5$:     MOV     #8$,R5          ;SET UP JSR RETURN ADDR
006570 000205                   RTS     R5              ;RELIEVE SUBROUTINE STATUS
006572 013700 012230    8$:     MOV     BUFBAS,R0       ;GET STARTING ADDR
006576 013701 012232            MOV     BUFSZE,R1       ;GET BUFFER SIZE
006602 123720 012120    10$:    CMPB    DXM00,(R0)+     ;CHECK A BYTE
006606 001404                   BEQ     15$             ;BRANCH IF NO ERROR
006610 004537 011272    12$:    JSR     R5,ERRR         ;REPORT AN ERROR
006614 000001                   .WORD   1               ;ERROR ID
006616 012306                   .WORD   ERR1            ;ADDR OF ERROR MESSAGE
006620 005301           15$:    DEC     R1              ;DECREMENT BUFFER COUNT
006622 001367                   BNE     10$             ;LOOP UNTIL ALL OF BUFFER CHECKED
006624 042777 000200 003226     BIC     #DONE,@DXCSA    ;TURN OFF FAST MODE
006632 042777 000004 003240     BIC     #SOSIEN,@DXESA  ;TURN OFF FAST MODE
006640 042777 000030 003212     BIC     #30,@DXCSA      ;CLEAR POSSIBLE CARRY IN EA BITS
006646 105737 012120            TSTB    DXM00           ;CHECK IF FIRST OF TWO PASSES
006652 100404                   BMI     30$             ;BR IF NOT FIRST PASS
006654 012737 100652 012120     MOV     #100652,DXM00   ;SET UP SECOND DATA PATTERN
006662 000721                   BR      GO1             ;DO TRANSFER WITH THIS DATA
```

```
1944  006664  004737  007030          30$:    JSR     PC,BUFCLR       ;PUT BUFFER BACK TO ZEROS
1945  006670  004537  010620          ENDT1:  JSR     R5,BUFAD        ;GET NEXT BUFFER ADDRESS
1946  006674  000402                          .WORD   402             ;BASE TO BE DIVISABLE BY 400
1947  006676  005701                          TST     R1              ;CHECK FOR OUT OF MEMORY
1948  006700  001702                          BEQ     LOOP1           ;BRANCH IF THERE IS MORE
1949  006702  000137  001410                  JMP     ENDTST          ;OTHERWISE WE'RE DONE WITH TEST
1950                                  ;
1951                                  ;  SUBROUTINE TO START THE DX IN SOSIEN MODE
1952                                  ;
1953  006706  013777  012120  003154  DXGO:   MOV     DXM00,@DXMOA    ;SEND DATA AND OP OUT TO MAINT REG
1954  006714  012577  003146                  MOV     (R5)+,@DXBCA    ;SET UP BYTE COUNT FROM PASSED PARM.
1955  006720  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
1956  006722  012746  006730                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
1957  006726  000002                          RTI                     ;;POP NEW PC AND PS
1958  006730                          64$:
1959  006730  052777  000103  003122          BIS     #103,@DXCSA     ;SET  FUNCTION TO IBM WRITE
1960                                                                  ;AND ENABLE INTERRUPTS
1961  006736  052777  060000  003124          BIS     #SELO!HLDO,@DXMOA       ;FORCE DX SELECTION
1962  006744  042777  060000  003116          BIC     #SELO!HLDO,@DXMOA       ;DROP SELECT LINES
1963  006752  052777  002000  003110          BIS     #CMDO,@DXMOA            ;RAISE COMMAND OUT
1964  006760  042777  002000  003102          BIC     #CMDO,@DXMOA            ;DROP COMMAND OUT
1965  006766  052777  000004  003104          BIS     #SOSIEN,@DXESA          ;ENABLE FAST SERVICE OUT/SERV IN
1966                                                                  ;TO UNLEASH NPR'S
1967  006774  005037  007026                  CLR     INTERR          ;CLEAR NO INTERRUPT OCCURED FLAG
1968  007000  005001                          CLR     R1
1969  007002  005301                  10$:    DEC     R1              ;WAIT HERE FOR INTERRUPT
1970  007004  001376                          BNE     10$
1971  007006  004537  011272                  JSR     R5,ERRR         ;ERROR, INTERRUPT NEVER  CAME
1972  007012  000002                          .WORD   2               ;ERROR IDENTIFER
1973  007014  012332                          .WORD   ERR2            ;ERROR MESSAGE POINTER
1974  007016  012737  000001  007026          MOV     #1,INTERR       ;SET NO INTERRUPT OCCURED FLAG
1975  007024  000205                          RTS     R5              ;CONTINUE TESTING
1976  007026  000000          INTERR: .WORD   0       ;SET TO 1 IF DX11 FAILED TO INTERRUPT
1977
1978
1979                                  ;       SUBROUTINE TO CLEAR THE CURRENT BUFFER
1980
1981  007030  013700  012230  BUFCLR: MOV     BUFBAS,R0       ;GET BUFFER START
1982  007034  013701  012232          MOV     BUFSZE,R1       ;GET BUFFER SIZE
1983  007040  105020          10$:    CLRB    (R0)+           ;CLEAR A BYTE
1984  007042  005301                  DEC     R1              ;COUNT IT
1985  007044  001375                  BNE     10$             ;LOOP TO CLEAR BUFFER
1986  007046  000207                  RTS     PC              ;RETURN TO CALLER
1987
```

```
                        .SBTTL  MODULE 3.1  TEST INITIALIZATION
                ;
                ;       MODULE 3.1      TEST SOFTWARE INITIALIZATION
                ;
                ;
                ;       CALLED AS FOLLOWS:
                ;
                ;       JSR     R5,INIT
                ;       RETURN COMES HERE
                ;
                ;
007050          INIT:
007050  012746  000340          MOV     #340,-(SP)      ;;PUT NEW PS ON STACK
007054  012746  007062          MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
007060  000002                  RTI                     ;;POP NEW PC AND PS
007062          64$:
007062  005037  012216          CLR     ERR             ;RESET ERROR OCCURRED INDICATION
007066  010537  012222          MOV     R5,LOCKAD       ;SET UP LOCK ON ERROR RETURN ADDR
007072  005037  007026          CLR     INTERR          ;CLEAR MISSED INTR FLAG
007076  000205                  RTS     R5
                ;
                ;       HARDWARE INITIALIZATION SUBROUTINE
                ;
                ;
007100  052777  000010  002772  DXRES:  BIS     #10,@DXESA      ;DISABLE DX TIME OUT, DEBUG ONLY
007106  042777  000200  002744          BIC     #DONE,@DXCSA    ;CLEAR DONE AND LOCKO
007114  042777  000006  002736          BIC     #6,@DXCSA       ;SET FUNCTION TO RESET
007122  052777  000001  002730          BIS     #1,@DXCSA       ;ISSUE THE DX RESET
007130  042777  001230  002722          BIC     #1230,@DXCSA    ;CLEAR POTENTIAL DISASTER OF ONLINE OR INTEN
                                                                ;CLEAR EA BITS ALSO
007136  012777  014000  002716          MOV     #SPW,@DXOSA     ;SET UP CU OFFSET TO SPW TABLE
007144  005737  012304                  TST     SPWMVD          ;IS SPW TABLE AT HOME BASE?
007150  001403                          BEQ     5$              ;BR IF YES
007152  012777  016000  002702          MOV     #SPW+2000,@DXOSA        ;SET UP ALTERNATE SPW LOCATION
007160  032777  100000  002706  5$:     BIT     #100000,@DXCBA  ;IS LOCKO ON?
007166  001402                          BEQ     10$             ;BR IF NO
007170  000005                          RESET                   ;WE'RE IN TROUBLE, DX DIDN'T RESET
007172  000742                          BR      DXRES           ;GO BACK AND MAKE SURE IT'S CLEARED
007174  000207                  10$:    RTS     PC              ;RETURN
```

```
                                        .SBTTL  MODULE 3.0.2  TST2  NPR ADDRESS UNIQUENESS
                                ;
                                ;       MODULE 3.0.2  NPR DATA ADDR UNIQUENESS TEST
                                ;
                                ;       THIS MODULE TESTS THE DX11'S ABILITY TO UNIQUELY ADDRESS EACH
                                ;       WORD OF MEMORY.  THIS TEST IS ACCOMPLISHED BY WRITING A BACKGROUND
                                ;       IN A BLOCK OF MEMORY.  A WORD IS THEN TRANSFERRED FROM THE DX TO THE
                                ;       MEMORY BLOCK.  THE CPU CHECKS THAT THE WORD ARRIVES AT ITS
                                ;       EXPECTED DESTINATION.  IF YES THE WORD IS RESTORED TO MATCH THE
                                ;       BACKGROUND, AND THE NEXT WORD IS TESTED.  THIS IS REPEATED
                                ;       FOR EACH WORD IN THE BLOCK AND FOR EACH BLOCK IN MEMORY.
                                ;
                                ;       NOTE:  THIS TEST MAY NOT DETECT ADDRESSING PROBLEMS
                                ;       WHICH ARE COMMON TO THE CPU AND DX11.  I.E. UNIBUS ADDRESSING PROBLEMS.
                                ;
007176  004537  007050          TST2:   JSR     R5,INIT         ;INITIALIZE THE SOFTWARE
007202  004737  007100                  JSR     PC,DXRES        ;INITIALIZE THE HARDWARE
007206  004537  010620                  JSR     R5,BUFAD        ;GET STARTING BUFFER ADDRESS
007212  101000                          .WORD   101000          ;PARM MEANS INIT AND SET UP1000 BYTE BUFFER
007214  013700  012160                  MOV     AVECT,R0        ;GET THE DX VECTOR ADDR
007220  012720  007306                  MOV     #INTR2,(R0)+    ;SET UP RETURN ADDR
007224  012710  000340                  MOV     #340,(R0)       ;SET INTR PSW
007230  012737  001000  012232  LOOP2:  MOV     #1000,BUFSZE    ;ESTABLISH BUFFER SIZE
007236  004737  007030                  JSR     PC,BUFCLR       ;CLEAR THE BUFFER
007242  013737  012230  012240          MOV     BUFBAS,DATAD    ;INITIALIZE DATA ADDR
007250  013737  012242  012244          MOV     BFBAPH,BFADPH   ;GET PHYSICAL BASE ADDR
007256  053777  012236  002574  LOP2:   BIS     EACUR,@DXCSA    ;SET EA BITS IN THE DX11
007264  012737  100377  012120          MOV     #100377,DXM00   ;DATA PATTERN TO BE WRITTEN BY DX

007272  013777  012244  002564          MOV     BFADPH,@DXBAA   ;SET UP DX DATA ADDRESS
007300  004537  006706                  JSR     R5,DXGO         ;START THE DX
007304  177776                          .WORD   -2              ;BYTE COUNT = 2

                                ;DX INTERRUPTS TO HERE AFTER NPR'ING A WORD
007306  005737  007026          INTR2:  TST     INTERR          ;DID WE COME HERE VIA DX INTR
007312  001006                          BNE     8$              ;BR IS NO,  STACK IS OK
007314  012716  007322                  MOV     #5$,@SP         ;SET UP RETURN ADDR
007320  000002                          RTI                     ;RELIEVE INTERRUPT STATUS
007322  012705  007330          5$:     MOV     #8$,R5          ;SET UP RETURN ADDR
007326  000205                          RTS     R5              ;RELIEVE SUBROUTINE STATUS
007330  027727  002704  177777  8$:     CMP     @DATAD,#-1      ;DID DATA GET WHERE IT WAS SUPPOSED TO?
                                                        ;ASSUME CPU CAN ADDRESS MEMORY CORRECTLY
007336  001404                          BEQ     10$             ;BRANCH IF OK
007340  004537  011272                  JSR     R5,ERRR         ;REPORT THE ERROR
007344  000003                          .WORD   3               ;ERROR NUMBER THREE
007346  012352                          .WORD   ERR3            ;ADDR OF ERROR MESSAGE
007350  005077  002664          10$:    CLR     @DATAD          ;RESTORE THE WORD UNDER TEST
007354  042777  000200  002476          BIC     #DONE,@DXCSA    ;TURN OFF INTERRUPTS
007362  042777  000004  002510          BIC     #SOSIEN,@DXESA  ;CLEAR FAST MODE
007370  042777  000030  002462          BIC     #30,@DXCSA      ;CLEAR POSSIBLE CARRY IN EA BITS
007376  062737  000002  012244          ADD     #2,BFADPH       ;BUMP THE PHYSICAL ADDRESS
007404  062737  000002  012240          ADD     #2,DATAD        ;BUMP DATA POINTER
007412  023737  012240  012234          CMP     DATAD,BUFMAX    ;DONE WITH BUFFER?
```

```
2084   007420  101716                  BLOS    LOP2            ;BRANCH IF NO
2085
2086   007422  004737  007030          JSR     PC,BUFCLR       ;RESTORE BUFFER
2087   007426  004537  010620          JSR     R5,BUFAD        ;GET NEXT BUFFER
2088   007432  001000                  .WORD   1000            ;SIZE =1000
2089   007434  005701                  TST     R1              ;DID WE GET ONE?
2090   007436  001674                  BEQ     LOOP2           ;BRANCH IF YES
2091   007440  000137  001410          JMP     ENDTST          ;OTHERWISE TERMINATE TEST
```

```
                                        .SBTTL          TST3  SPW ADDRESSING
                                ;
                                ;       MODULE 3.0.3    SPW ADDRESSING TEST
                                ;
                                ;
                                ;       THIS MODULE TESTS THE ABILITY OF THE DX TO ACCESS THE STATUS
                                ;       POINTER WORD (SPW) TABLE IN ALL POSSIBLE LOCATIONS IN MEMORY.
                                ;       A BASE ADDRESS FOR THE SPW IS GENERATED.
                                ;       THIS SPW TABLE IS FILLED WITH A PATTERN OF 000DEV
                                ;       WHERE DEV IS EQUAL TO THE DISPLACEMENT OF THE WORD FROM THE
                                ;       SPW BASE; I.E.
                                ;       SPW BASE    =000000
                                ;       SPW BASE +1 =000001
                                ;                 ''
                                ;                ''
                                ;                 ''
                                ;                 ''
                                ;       SPW BASE+255=000377
                                ;
                                ;       GIVING A TOTAL OF 256 WORDS FOR THE SPW TABLE WITH IMMEDIATE
                                ;       STATUS EQUAL TO THE DEVICE NUMBER.
                                ;       AN ISS SEQUENCE IS INITIATED WITH A DEVICE # OF 0.  IF
                                ;       ADRECC AND ADRECD DON'T COME ON FOR THIS DEV, NO TESTING IS DONE,
                                ;       THE DEV # IS INCREMENTED, AND ISS IS DONE AGAIN.
                                ;       IF ADRECC AND ADRECD COME ON IT MEANS THE DX JUMPERS ARE CUT TO
                                ;       RESPOND TO THIS DEV NUMBER. THE ISS SEQUENCE IS THEN CONTINUED
                                ;       UNTIL THE STATUS IS IN THE DXOS REG.  THIS STATUS IS THEN
                                ;       CHECKED AND SHOULD BE EQUAL TO THE DEV # IF THE SPW WAS
                                ;       ACCESSED CORRECTLY.
                                ;
                                ;       THIS PROCEDURE IS REPEATED FOR ALL 256 POSSIBLE DEVICES AT
                                ;       ALL POSSIBLE LOCATIONS FOR THE SPW.
                                ;
                                ;
                                ;       THE FIRST RECOGNIZED  DEV NO. AND COUNT OF THE FIRST
                                ;       GROUP OF RECOGNIZED DEVICES IS SAVED AND PRINTED AT END
                                ;       PASS TIME.  IF NO DEV IS RECOGNIZED AND ERROR MESSAGE
                                ;       IS OUTPUT AT THE END OF THE TEST.
                                ;
007444  004537  007050          TST3:   JSR     R5,INIT         ;INITIALIZE THE SOFTWARE
007450  005037  012264                  CLR     RECORD          ;SET UP TO RECORD IBM DEV RECOG.
007454  005037  012266                  CLR     FRSTD   ;SET TO RECORD FIRST DEV RECOGNIZED
007460  005037  012272                  CLR     DEVCT           ;CLEAR DEVICE COUNTER
007464  004737  007100                  JSR     PC,DXRES        ;RESET THE DX
007470  012737  000400  012252          MOV     #400,CMD        ;INITIALIZE A COMMAND OF ZERO
007476  004537  010620                  JSR     R5,BUFAD        ;GET THE FIRST BUFFER ADDR
007502  102000                          .WORD   102000          ;SPW TO BE ON 2000 BOUNDARY
007504  012737  002000  012232          MOV     #2000,BUFSZE    ;SET UP BUFFER SIZE
007512  004737  007030                  JSR     PC,BUFCLR       ;CLEAR BUFFER
007516  005037  012250          LOOP3:  CLR     DEV             ;START AT DEVICE 0
007522  013700  012230                  MOV     BUFBAS,R0       ;GET SPW TABLE BASE
007526  013720  012250          10$:    MOV     DEV,(R0)+       ;MOV IMMED. STATUS TO SPW EQUAL TO DEV
007532  005237  012250                  INC     DEV             ;INC TO NEXT STATUS = NEXT DEV
007536  022737  000377  012250          CMP     #255.,DEV       ;IS DST FULL?
```

```
2148  007544  103370                        BHIS    10$             ;DO MORE IF IT IS NOT
2149  007546  005037  012250                CLR     DEV             ;START AT DEV 0
2150  007552  013777  012242  002302  LOP3: MOV     BFBAPH,@DXOSA   ;SET UP OFFSET FOR SPW
2151  007560  053777  012236  002272        BIS     EACUR,@DXCSA    ;SET UP EXT ADDR BITS
2152  007566  004537  010034                JSR     R5,PTYGEN       ;ASSIGN PARITY TO THE DEVICE BYTE
2153  007572  012250                        .WORD   DEV             ;ADDR OF DATA TO ASSIGN PARITY TO
2154  007574  113777  012250  002254        MOVB    DEV,@DXCAA      ;SET DEV TO CUAR
2155  007602  053777  012250  002260        BIS     DEV,@DXMOA      ;PUT DEV ADDR ON BUSOUT
2156  007610  052777  004000  002252        BIS     #ADRO,@DXMOA    ;RAISE ADDR OUT
2157  007616  032777  000001  002250        BIT     #1,@DXCBA       ;IS  ADRECD ON?
2158  007624  001452                        BEQ     20$             ;BR IF ADDR NOT RECOGNIZED
2159  007626  032777  000002  002240        BIT     #2,@DXCBA       ;IS ADRECC ON?
2160  007634  001446                        BEQ     20$             ;BR IF ADDR NOT RECOGNIZED
2161  007636  004537  010074                JSR     R5,SAVDEV       ;GO SAVE RECOGNIZED DEV DATA
2162  007642  052777  060000  002220        BIS     #HLDO!SELO,@DXMOA       ;RAISE SEL OUT, HOLD OUT
2163  007650  042777  004000  002212        BIC     #ADRO,@DXMOA    ;DROP ADRO
2164  007656  043777  012250  002204        BIC     DEV,@DXMOA      ;REMOVE DEVICE FROM BUSO
2165  007664  053777  012252  002176        BIS     CMD,@DXMOA      ;PUT CMD =0 ON BUSOUT
2166  007672  052777  002000  002170        BIS     #CMDO,@DXMOA    ;RAISE CMD OUT
2167  007700  043777  012252  002162        BIC     CMD,@DXMOA      ;REMOVE CMD FROM BUSO
2168  007706  042777  002000  002154        BIC     #CMDO,@DXMOA    ;DROP CMD OUT
2169  007714  123777  012250  002140        CMPB    DEV,@DXOSA      ;IS STATUS EQUAL TO DEV?
2170  007722  001404                        BEQ     10$             ;BR IF OK
2171  007724  004537  011272                JSR     R5,ERRR         ;REPORT ERROR
2172  007730  000004                        .WORD   4               ;ERROR ID
2173  007732  012411                        .WORD   ERR4            ;ERROR MESSAGE ADDR
2174  007734  052777  001000  002126  10$:  BIS     #SRVO,@DXMOA    ;RAISE SERVICE OUT TO RELIEVE STATUS
2175  007742  042777  001000  002120        BIC     #SRVO,@DXMOA    ;DROP SERVICE OUT
2176  007750  000402                        BR      30$
2177  007752  004537  010140          20$:  JSR     R5,DONTSV       ;GO HANDLE RECOGNIZED DEV DATA
2178  007756  004737  007100          30$:  JSR     PC,DXRES        ;RESTORE DX
2179  007762  005237  012250                INC     DEV             ;GO TO NEXT DEV
2180  007766  105737  012250                TSTB    DEV             ;LAST DEV?
2181  007772  001267                        BNE     LOP3            ;BR IF MORE DEVICES
2182  007774  004737  007030                JSR     PC,BUFCLR       ;CLEAR OLD BUFFER BEFORE GETTING NEW ONE
2183  010000  004537  010620                JSR     R5,BUFAD        ;GET NEXT BUFFER BASE
2184  010004  002000                        .WORD   2000            ;DIVISABLE BY 2000
2185  010006  005701                        TST     R1              ;DID WE GET ANOTHER BUFFER?
2186  010010  001642                        BEQ     LOOP3           ;BR IF YES AND DO NEXT ONE
2187  010012  005737  012272                TST     DEVCT           ;TEST IS DONE, DID WE EVER FIND A
2188                                                                ;RECOGNIZED DEVICE?
2189  010016  001004                        BNE     40$             ;BR IF YES
2190  010020  004537  011272                JSR     R5,ERRR         ;ERROR, NO ADREDD AND ADRECD FOR ANY
2191                                                                ;IBM DEVICE NUMBER
2192  010024  000005                        .WORD   5
2193  010026  012440                        .WORD   ERR5            ;ADDR OF ERROR MSG
2194  010030  000137  001410          40$:  JMP     ENDTST          ;TERMINATE TEST
2195
2196
2197
2198
2199
2200                                  ;       PARITY ASSIGNMENT SUBROUTINE
2201                                  ;
2202                                  ;       CALLED AS FOLLOWS:
2203                                  ;       JSR     R5,PTYGEN
```

```
2204                                    ;       .WORD   ADDR            ;ADDRESS OF DATA TO WHICH PARITY WILL BE
2205                                                                    ;ASSIGNED
2206                                    ;       RETURN HERE
2207                                    ;
2208                                    ;       REGISTERS R2, R3, R4 ARE DISTROYED BY THIS SUBROUTINE
2209                                    ;
2210                                    ;
2211  010034  012503                   PTYGEN: MOV     (R5)+,R3        ;GET ADDR OF DATA
2212  010036  011304                           MOV     (R3),R4         ;GET THE DATA
2213  010040  005002                           CLR     R2              ;INITIALIZE THE PARTIY SW
2214  010042  106304                   PTY2:   ASLB    R4              ;CHECK A BIT
2215  010044  102001                           BVC     PTY3            ;BR IF ONES
2216  010046  005102                           COM     R2              ;OTHERWISE FLIP THE PARITY SW
2217  010050  106304                   PTY3:   ASLB    R4              ;TWO BITS GET CHECKED AT ONCE
2218  010052  001373                           BNE     PTY2            ;IF R4 = 0 WE'D BE DONE
2219  010054  005702                           TST     R2              ;DID THE SWITCH END UP SET?
2220  010056  100403                           BMI     PTY4            ;BR IF NO
2221  010060  052713  000400                   BIS     #400,(R3)       ;SET THE BIT IN THE TARGET DATA
2222  010064  000402                           BR      PTY5            ;EXIT
2223  010066  042713  000400           PTY4:   BIC     #400,(R3)       ;CLEAR THE BIT IN THE TARGET DATA
2224  010072  000205                   PTY5:   RTS     R5              ;RETURN
2225
2226
2227
2228  010074  005737  012264           SAVDEV: TST     RECORD          ;ARE WE RECORDING DEVICES FOUND?
2229  010100  001016                           BNE     20$             ;BR IF NO AND EXIT
2230  010102  005737  012266                   TST     FRSTD           ;IS THIS THE VERY FIRST DEV RECOGNIZED?
2231  010106  001011                           BNE     10$             ;BR IF NO
2232  010110  052737  000001  012266           BIS     #1,FRSTD        ;SET FIRST DEV FOUND SW
2233  010116  113737  012250  012270           MOVB    DEV,FRSTDV      ;SAVE FIRST DEVICE FOUND
2234  010124  012737  000001  012302           MOV     #1,DEVFND       ;SET SWITCH = DEVICE FOUND
2235  010132  005237  012272           10$:    INC     DEVCT           ;COUNT THE DEV
2236  010136  000205                   20$:    RTS     R5
2237
2238
2239
2240
2241  010140  005737  012272           DONTSV: TST     DEVCT           ;HAVE WE FOUND ANY DEV YET?
2242  010144  001403                           BEQ     30$             ;BR IF NO
2243  010146  052737  000001  012264           BIS     #1,RECORD       ;TURN OFF DEV RECORDING
2244  010154  000205                   30$:    RTS     R5
2245
2246
2247
```

```
                                        .SBTTL  MODULE 3.0.4   TST4  DST ACCESS TEST
                                ;
                                ;       MODULE 3.0.4    DEVICE STATUS TABLE (DST) ACCESS TEST
                                ;
                                ;       THIS TEST CHECKS THE ABILITY TO ACCESS ALL BYTES OF THE
                                ;       DST IN ALL AREAS OF MEMORY.  THE OBJECTIVE IS TO CHECK LOADING
                                ;       OF THE DXBA FROM THE SPW DST BASE, AND COMMAND (CUCR)
                                ;
                                ;       A VALID IBM DEVICE FROM TST3 IS CHECKED.  THEN A BASE ADDR
                                ;       IS GENERATED FOR THE DST.  EACH BYTE OF THE DST IS THEN LOADED
                                ;       WITH ITS OFFSET FROM THE DST BASE.
                                ;
                                ;       A SPW ENTRY FOR THE VALID IBM DEVICE IS GENERATED WITH AP-
                                ;       PROPRIATE DST BASE AND ZERO IMMEDIATE STATUS TO ASSURE A DST
                                ;       ACCESS.  AN ISS SEQUENCE IS DONE WITH THE VALID DEV AND
                                ;       EVERY POSSIBLE IBM COMMAND THEREBY ACCESSING EVERY LOCATION
                                ;       IN THE DST.  AFTER THE ISS, THE STATUS WILL EQUAL THE COMMAND
                                ;       OR AND ERROR IS FLAGGED.
                                ;
010156  004537  007050    TST4:   JSR     R5,INIT         ;INITIALIZE THE SOFTWARE
010162  004737  007100            JSR     PC,DXRES        ;INITIALIZE THE HARDWARE
010166  005737  012302            TST     DEVFND          ;DO WE HAVE A DEVICE?
010172  001006                    BNE     10$             ;BR IF YES
010174  004537  005372            JSR     R5,PRINT        ;TEST 3 MUST BE RUN SUCCESSFULLY FIRST
010200  012535                    .WORD   T4MSG           ;TO FIND AND IBM DEVICE #
010202  000000                    .WORD   0               ;AND ASSURE GOOD SPW ACCESS CAPABILITY
010204  000137  010614            JMP     END4            ;ABORT TEST
010210  005037  012250    10$:    CLR     DEV             ;CLEAN UP CURRENT DEV NO.
010214  113737  012270  012250    MOVB    FRSTDV,DEV      ;GET THE VALID DEV
010222  004537  010034            JSR     R5,PTYGEN       ;ASSIGN PARITY TO IT
010226  012250                    .WORD   DEV
010230  004537  010620            JSR     R5,BUFAD        ;GET A BUFFER OF 256 BYTES
010234  100400                    .WORD   100400
010236  012737  000400  012232    MOV     #400,BUFSZE     ;SET BUFFER SIZE
010244  004737  007030            JSR     PC,BUFCLR       ;CLEAR THE BUFFER
010250  005000            LOOP4:  CLR     R0              ;START WITH CMD = 0
010252  013701  012230            MOV     BUFBAS,R1       ;GET THE DST BASE
010256  110021            10$:    MOVB    R0,(R1)+        ;SET CMD IN DST
010260  005200                    INC     R0              ;NEXT CMD
010262  022700  000377            CMP     #255.,R0        ;ARE ALL STATUS = TO CMD?
010266  103373                    BHIS    10$             ;DO 256 BYTES
010270  005000                    CLR     R0
010272  013700  012250            MOV     DEV,R0          ;GET DEVICE
010276  042700  177400            BIC     #177400,R0
010302  006300                    ASL     R0              ;MAKE IT A WORD OFFSET
010304  017701  001552            MOV     @DXOSA,R1       ;GET THE SPW BASE
010310  042701  001777            BIC     #1777,R1        ;CLEAN IT UP
010314  050100                    BIS     R1,R0           ;ADD DEV TO OFFSET
010316  005737  012172            TST     KT              ;MM IN USE?
010322  001412                    BEQ     15$             ;BR IF NO
010324  013701  012236            MOV     EACUR,R1        ;GET EXT ADDR BITS
010330  000301                    SWAB    R1              ;POSITION FOR USE IN PAR
010332  006201                    ASR     R1
```

```
2304  010334  010137  172352                 MOV     R1,@#KIPAR5     ;USE PAR5 TO GET TO SPW TABLE
2305  010340  042700  040000                 BIC     #40000,R0       ;SET SPW ADDR TO USE PAR5
2306  010344  052700  120000                 BIS     #120000,R0
2307  010350  013701  012242          15$:   MOV     BFBAPH,R1       ;SET PHY BUFFER BASE
2308  010354  105001                         CLRB    R1              ;IMMED STATUS = 0
2309  010356  010110                         MOV     R1,(R0)         ;SET UP SPW
2310  010360  005037  012252                 CLR     CMD             ;START WITH CMD = 0
2311  010364  004537  010034          LOP4:  JSR     R5,PTYGEN       ;GET COMMAND PARITY
2312  010370  012252                         .WORD   CMD
2313  010372  053777  012236  001460         BIS     EACUR,@DXCSA    ;SET UP EXT ADDR BITS
2314  010400  113777  012250  001450         MOVB    DEV,@DXCAA      ;SET DEVICE TO CUAR
2315  010406  053777  012250  001454         BIS     DEV,@DXMOA      ;PUT DEV ADDR ON BUS OUT
2316  010414  052777  004000  001446         BIS     #ADRO,@DXMOA    ;ADDR OUT
2317  010422  052777  060000  001440         BIS     #HLDO!SELO,@DXMOA       ;SELECT & HOLD OUT
2318  010430  042777  004000  001432         BIC     #ADRO,@DXMOA    ;DROP ADDR OUT
2319  010436  043777  012250  001424         BIC     DEV,@DXMOA      ;DROP DEV FROM BUS
2320  010444  053777  012252  001416         BIS     CMD,@DXMOA      ;COMMAND TO BUS
2321  010452  052777  002000  001410         BIS     #CMDO,@DXMOA    ;COMMAND OUT
2322  010460  043777  012252  001402         BIC     CMD,@DXMOA      ;CLR CMD FROM BUS
2323  010466  042777  002000  001374         BIC     #CMDO,@DXMOA    ;DROP COMMAND OUT
2324  010474  122737  000004  012252         CMPB    #4,CMD          ;IS IT A SENSE COMMAND?
2325  010502  001004                         BNE     5$              ;BR IF NO
2326  010504  105777  001352                 TSTB    @DXOSA          ;STATUS FORCED TO ZERO ON SENSE CMD
2327  010510  001411                         BEQ     10$             ;BR IF GOOD
2328  010512  000404                         BR      7$              ;REPORT ERROR
2329  010514  123777  012252  001340   5$:   CMPB    CMD,@DXOSA      ;IS STATUS EQUAL DEVICE?
2330  010522  001404                         BEQ     10$             ;BR IF YS
2331  010524  004537  011272          7$:    JSR     R5,ERRR         ;ERROR IN DST ACCESS
2332  010530  000006                         .WORD   6               ;ERROR #
2333  010532  012514                         .WORD   ERR6            ;MESSAGE ADDR
2334  010534  052777  001000  001326  10$:   BIS     #SRVO,@DXMOA    ;RELIEVE STATUS
2335  010542  042777  001000  001320         BIC     #SRVO,@DXMOA
2336  010550  004737  007100                 JSR     PC,DXRES        ;CLEAR THE DX
2337  010554  005237  012252                 INC     CMD             ;TRY NEXT COMMAND
2338  010560  105737  012252                 TSTB    CMD             ;DONE IF BACK TO ZERO
2339  010564  001277                         BNE     LOP4            ;BR TO DO NEXT CMD
2340  010566  004737  007030                 JSR     PC,BUFCLR       ;CLEAR OLD BUFFER
2341  010572  004537  010620          20$:   JSR     R5,BUFAD        ;GET NEXT BUFFER BASE
2342  010576  000400                         .WORD   400
2343  010600  005701                         TST     R1              ;DID WE GET A BUFFER?
2344  010602  001004                         BNE     END4            ;BR IF NO,END TEST
2345  010604  005737  012242                 TST     BFBAPH          ;IS BUFFER BASE 0'S?
2346  010610  001770                         BEQ     20$             ;BR IF YES, THE DX DOES NOT DO A
2347                                                                 ;DST FETCH IF SPW ENTRY IS ALL ZEROS
2348  010612  000616                         BR      LOOP4           ;DO NEXT BUFFER
2349  010614  000137  001410          END4:  JMP     ENDTST          ;END OF TEST 4
```

```
                                        .SBTTL  BASE ADDRESS GENERATOR MODULE
                                ;       MODULE 3.2    BASE ADDRESS GENERATOR
                                ;
                                ;       THIS MODULE SETS UP THE STaRTING ADDRESS FOR DX BUFFERS
                                ;       AND HAS RESPONSIBILITY FOR ESTABLISHING EXTENDED ADDR
                                ;       BITS,BUFFER BASE, BUFFER SIZE, AND HAS CONTROL OF
                                ;       THE MEMORY MGT. REGISTERS.  KIPAR6 IS ALWAYS USED
                                ;       TO ACCESS THE DXBUFFERS IF MEM MGT. IS ENABLED.
                                ;
                                ;       THIS MODULE IS CALLED AS FOLLOWS:
                                ;
                                ;       JSR     R5,BUFAD
                                ;       .WORD   PARM
                                ;       RETURN HERE WITH R1=0 MEANS BUFFER ALLOCATED
                                ;                         R1=1 MEANS NO MEMORY LEFT
                                ;       PARM IS DEFINED AS FOLLOWS:
                                ;       BIT 15=1          INITIALIZE  TO FIRST BUFFER LOCATION
                                ;       BIT 15=0          GET NEXT BUFFER LOCATION
                                ;       BITS 14:0         BASE ADDR OF BUFFER MUST BE DIVISABLE BY
                                ;                         THIS NUMBER
010620  012500                  BUFAD:  MOV     (R5)+,R0        ;GET THE PASSED PARAMETER
010622  005001                          CLR     R1              ;RESET "OUT OF MEMORY" INDICATOR
010624  005700                          TST     R0              ;CHECK FOR INITIALIZE SW
010626  100050                          BPL     NEXTB           ;BRANCH IF NOT TO INITIALIZE
010630  005037  012236                  CLR     EACUR           ;START INIT PROCESS BY CLEARING
                                                                ;THE EA BITS
010634  012702  016000                  MOV     #ENDALL,R2      ;GET VIRTUAL END OF PROGRAM ADDR
010640  042700  100000                  BIC     #100000,R0      ;CLEAR HIGH BIT OF PARAM
010644  060002                          ADD     R0,R2           ;OFFSET ABOVE PROGRAM
010646  162700  000001                  SUB     #1,R0           ;DEVELOP A MASK
010652  040002                          BIC     R0,R2           ;TO MAKE BASE DIVISABLE BY PROPER AMT.
010654  010237  012230                  MOV     R2,BUFBAS       ;SAVE VIRTUAL BUFFER BASE
010660  010237  012242                  MOV     R2,BFBAPH       ;SAVE PHYSICAL BUFFER BASE, SAME AS VIRT
                                                                ;AT THIS POINT.
010664  005737  012172                  TST     KT              ;IS KT IN USE?
010670  001421                          BEQ     ENDB            ;BRANCH IF NO TO END OF MODULE
010672  012703  000006                  MOV     #6,R3           ;INIT LOOP COUNTER
010676  005037  172354                  CLR     @#KIPAR6        ;START TO INITIALIZE KIPAR6
010702  006202                  10$:    ASR     R2      ;SHIFT ADDRESS RIGHT 6 PLACES
010704  077302                          SOB     R3,10$          ;TO ALLIGN WITH PAR
010706  010237  172354                  MOV     R2,@#KIPAR6     ;SET UP PAR6 TO REFERENCE FIRST BUFFER
010712  042737  177700  012230          BIC     #177700,BUFBAS  ;CLEAR UP VIRTUAL BUFFER ADDR
010720  052737  140000  012230          BIS     #140000,BUFBAS  ;BUFFER ACCESS IS ALWAYS THRU PAR6
010726  012737  000001  177572          MOV     #1,@#SR0        ;TURN ON MM
010734  063700  012230          ENDB:   ADD     BUFBAS,R0       ;FIND TOP OF CURRENT BUFFER
010740  005200                          INC     R0      ;
010742  010037  012234                  MOV     R0,BUFMAX       ;SAVE VIRT END OF BUFFER
010746  000550                          BR      ENDB1           ;DONE WITH INITIALIZATION
                                ;
                                ;
                                ;       COME HERE IF NEXT BUFFER ADDR IS WANTED
                                ;
010750  060037  012230          NEXTB:  ADD     R0,BUFBAS       ;INCR TO NEXT BUFFER SPACE
010754  042737  000077  012230          BIC     #77,BUFBAS      ;CLEAR POSSIBLE LOW ORDER BITS
010762  060037  012242                  ADD     R0,BFBAPH       ;STEP PHYSICAL ADDR
010766  042737  000077  012242          BIC     #77,BFBAPH      ;CLEAR POSSIBLE LOW ORDER BITS
```

```
2406  010774  103007                        BCC     2$              ;BR IF NO CARRY
2407  010776  062737  000010  012236        ADD     #10,EACUR       ;STEP EXTENDED ADDR BITS
2408  011004  023737  012236  012224        CMP     EACUR,EABITS    ;HAVE THE EXT ADR BITS GONE TOO HIGH?
2409  011012  101125                        BHI     30$             ;BR IF YES TO SET ALL DONE SW AND EXIT
2410  011014  005737  012172          2$:   TST     KT              ;IS MEM MNG IN USE?
2411  011020  001511                        BEQ     20$             ;BRANCH IF NO
2412  011022  032737  020000  012230        BIT     #20000,BUFBAS   ;WAS THERE A CARRY TO ACCESS NEXT PAR?
2413  011030  001417                        BEQ     10$             ;BR IF NO CARRY
2414  011032  062737  000200  172354        ADD     #200,@#KIPAR6   ;TAKE A 4K STEP
2415  011040  023737  172354  012170        CMP     @#KIPAR6,EASIZE ;CHECK FOR TOO HIGH
2416  011046  103402                        BLO     5$              ;BRANCH IF OK
2417  011050  005201                        INC     R1              ;SET THE "ALL DONE" SWITCH
2418  011052  000506                        BR      ENDB1           ;EXIT
2419  011054  042737  177700  012230  5$:   BIC     #177700,BUFBAS  ;START BUFFER BASE OVER
2420  011062  052737  140000  012230        BIS     #140000,BUFBAS  ;ACCESS THROUGH PAR6
2421  011070  013703  012230          10$:  MOV     BUFBAS,R3       ;GET THE BASE ADDR
2422  011074  060003                        ADD     R0,R3           ;CALCULATE TOP OF BUFFER
2423  011076  032703  020000                BIT     #20000,R3       ;DIT IT CARRY ACROSS A PAR BOUNDARY
2424  011102  001322                        BNE     NEXTB           ;BR IS YES.  GO BACK AND GET A BETTER ADDR
2425  011104  010337  012234                MOV     R3,BUFMAX       ;SAVE TOP OF BUFFER ADDR
2426  011110  013703  012242                MOV     BFBAPH,R3       ;GET PHYSICAL BUF BASE
2427  011114  060003                        ADD     R0,R3           ;ADD BUFFER SIZE TO GET BUF TOP
2428  011116  023737  012236  012224        CMP     EACUR,EABITS    ;ARE WE IN THE TOP PHYSICAL BANK?
2429  011124  001006                        BNE     12$             ;BR IF NO
2430  011126  020337  012226                CMP     R3,PHYMAX       ;ARE WE OVER THE TOP OF MEM?
2431  011132  103055                        BHIS    30$             ;BR IF YES, SET DONE SW
2432  011134  020327  000004                CMP     R3,#4           ;WAS THERE A CARRY OUT OF TOP BANK?
2433  011140  101452                        BLOS    30$             ;BR IF YES, WE MUST BE DONE
2434  011142  013703  012242          12$:  MOV     BFBAPH,R3       ;GET PHYSICAL BUFFER BASE
2435  011146  060003                        ADD     R0,R3           ;FIND TOP OF PHY BUFFER
2436  011150  103677                        BCS     NEXTB           ;BR IF CARRY OVER 32K BNDRY
2437                                                                ;AND GO BACK AND GET A BUFFER WHICH DOESN'T
2438                                                                ;STRADDEL A 32K BOUNDARY
2439  011152  005737  012236                TST     EACUR           ;ARE WE IN THE FIRST BANK?
2440  011156  001010                        BNE     3$              ;BR IF NO
2441  011160  020337  012164                CMP     R3,MSIZE        ;ARE WE IN THE LOADER AREA?
2442  011164  101405                        BLOS    3$              ;BR IF OK
2443  011166  023737  012242  012166        CMP     BFBAPH,MMAX     ;IS BUFFER ABOVE LOADERS?
2444  011174  103001                        BHIS    3$              ;BRANCH IF WE'RE OK
2445  011176  000664                        BR      NEXTB           ;GO BACK AND GET A BETTER BUFFER ADDR
2446  011200  012777  014000  000654  3$:   MOV     #SPW,@DXOSA     ;SET UP SPW ADDR
2447  011206  005037  012304                CLR     SPWMVD          ;CLEAR SPW BASE ADDR MOVED SWITCH
2448  011212  020327  014000                CMP     R3,#SPW         ;IS TOP OF BUFFER IN SPW?
2449  011216  101411                        BLOS    15$             ;BR IF OK
2450  011220  023727  012242  016000        CMP     BFBAPH,#SPW+2000        ;IS BUFFER BASE IN TT?
2451  011226  103005                        BHIS    15$             ;BR IF NO
2452  011230  012777  016000  000624        MOV     #SPW+2000,@DXOSA        ;MOV THE SPW
2453  011236  005237  012304                INC     SPWMVD          ;SET SPW ADDR MOVED TO ALTERNATE SWITCH
2454  011242  000412                  15$:  BR      ENDB1           ;GO TO END OF MODULE
2455  011244  013737  012230  012234  20$:   MOV     BUFBAS,BUFMAX   ;START TO GET BUFMAXIMUM
2456  011252  060037  012234                ADD     R0,BUFMAX       ;ADD IN BUFFER SIZE
2457  011256  023737  012234  012164        CMP     BUFMAX,MSIZE    ;SEE IF WE'RE OUT OF MEMORY
2458  011264  101401                        BLOS    ENDB1           ;BR IF OK
2459  011266  005201                  30$:  INC     R1              ;SET OUT OF MEMORY SWITCH
2460  011270  000205                  ENDB1: RTS    R5              ;RETURN FROM MODULE
```

```
                                                .SBTTL  MODULE 3.3      REPORT ERRORS
                                        ;       MODULE 3.3      ERROR REPORTING
                                        ;
                                        ;       CALLED AS FOLLOWS:
                                        ;
                                        ;       JSR     R5,ERRR
                                        ;       .WORD   X
                                        ;       .WORD   Y
                                        ;       RETURN (UNLESS LOCK ON ERROR FUNCTION OVERRIDES RETURN)
                                        ;       X EQUALS ERROR IDENTIFICATION NUMBER
                                        ;       Y EQUALS ADDR OF UNIQUE ERROR MESSAGE
                                        ;
                                        ;       THIS MODULE FORMATS AND PRINTS ERROR MESSAGES AND HANDLES
                                        ;       ERROR OPTIONS SUCH AS INHIBIT ERROR MESSAGES, SHORT ERROR
                                        ;       REPORT, SCOPE LOOP, HALT ON ERROR, ETC.
                                        ;
011272  032777  100000  167640  ERRR:   BIT     #SW15,@SWR      ;HALT ON ERROR ON?
011300  001401                          BEQ     5$              ;BR IF NO
011302  000000                          HALT                    ;SW15 ON AND ERROR OCCURRED.
011304  013737  012206  012216  5$:     MOV     CTESTN,ERR      ;SET ERROR DETECTED SW
011312  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
011314  012746  011322                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
011320  000002                          RTI                     ;;POP NEW PC AND PS
011322                          64$:
011322  005737  012220                  TST     LOCK            ;ARE WE LOCKED ON ERROR?
011326  001407                          BEQ     10$             ;BR IF NO
011330  032777  040000  167602          BIT     #SW14,@SWR      ;SCOPE LOOP SW ON?
011336  001413                          BEQ     15$             ;BR IF NO
011340  013705  012222                  MOV     LOCKAD,R5       ;GET THE LOOP ADDR
011344  000205                          RTS     R5              ;GO TO IT
011346  032777  040000  167564  10$:     BIT     #SW14,@SWR      ;SCOPE LOOP ON?
011354  001404                          BEQ     15$             ;BR IF NO
011356  013737  012206  012220          MOV     CTESTN,LOCK     ;SET LOCKED ON ERR SW
011364  000402                          BR      20$
011366  005037  012220          15$:    CLR     LOCK            ;TURN OF LOCKED SW
011372  010537  012254          20$:    MOV     R5,ERRPC        ;SAVE ERR PC
011376  012537  012256                  MOV     (R5)+,ERRID     ;SAVE ERROR ID
011402  012537  012260                  MOV     (R5)+,ERRMS     ;SAVE ERROR MESSAGE ADDR
011406  032777  020000  167524          BIT     #SW13,@SWR      ;INHIBIT PRINT ON?
011414  001402                          BEQ     25$             ;BR IF NO
011416  000137  012040                  JMP     97$             ;TERMINATE MODULE
011422  004537  005372          25$:    JSR     R5,PRINT        ;PRINT ERRPC
011426  013307                          .WORD   ERPC            ;ADDR OF MESSAGE
011430  000000                          .WORD   0               ;NO SPECIAL PRINT CONTROLS
011432  162737  000004  012254          SUB     #4,ERRPC        ;BACKUP ADDR TO CORRECT VALUE
011440  004537  005372                  JSR     R5,PRINT        ;PRINT THE PC NUMBER ITSSELF
011444  012254                          .WORD   ERRPC
011446  100001                          .WORD   100001          ;NO CRLF, CONVERT TO ASCII
011450  004537  005372                  JSR     R5,PRINT        ;PRINT ERROR NUMBER ON SAME LINE
011454  013322                          .WORD   ERNO
011456  000001                          .WORD   1               ;INHIBIT CRLF
011460  004537  005372                  JSR     R5,PRINT        ;PRINT THE OCT
011464  012256                          .WORD   ERRID
011466  100001                          .WORD   100001
```

```
2517  011470  013737  012260  011502          MOV     ERRMS,30$       ;GET MESSAGE POINTER
2518  011476  004537  005372                  JSR     R5,PRINT        ;PRINT THE UNIQUE MESSAGE
2519  011502  000000                  30$:    .WORD   0               ;FILLED FROM TWO LINES UP
2520  011504  000000                          .WORD   0               ;NO SPECIAL CONTROLS
2521  011506  032777  010000  167424          BIT     #SW12,@SWR      ;SHORT ERROR REPORT ON?
2522  011514  001145                          BNE     95$             ;BR IF YES, WE'RE DONE
2523  011516  004537  005372                  JSR     R5,PRINT        ;PRINT TEST NO.
2524  011522  013336                          .WORD   ERTN
2525  011524  000000                          .WORD   0
2526  011526  004537  005372                  JSR     R5,PRINT        ;PRINT THE OCTAL TEST NO.
2527  011532  012206                          .WORD   CTESTN
2528  011534  100001                          .WORD   100001
2529  011536  022737  000005  012256          CMP     #5,ERRID        ;IS IT ERROR #5?
2530  011544  001531                          BEQ     95$             ;BR IF YES, REST OF STUFF IS MEANINGLESS
2531  011546  004537  005372                  JSR     R5,PRINT        ;PRINT VIRT ADDR
2532  011552  013351                          .WORD   ERADR
2533  011554  000000                          .WORD   0
2534  011556  004537  005372                  JSR     R5,PRINT        ;PRINT VIRT ADDR OCTAL
2535  011562  012230                          .WORD   BUFBAS
2536  011564  100001                          .WORD   100001
2537  011566  005737  012172                  TST     KT              ;IS MEM MGT IN USE?
2538  011572  001431                          BEQ     90$             ;BR IF NO WE'RE DONE
2539  011574  004537  005372                  JSR     R5,PRINT        ;PRINT PHYSICAL ADDR
2540  011600  013404                          .WORD   ERADPH
2541  011602  000000                          .WORD   0
2542  011604  004537  005372                  JSR     R5,PRINT        ;PRINT PHYSICAL ADDR OCTAL
2543  011610  012242                          .WORD   BFBAPH
2544  011612  100001                          .WORD   100001
2545  011614  004537  005372                  JSR     R5,PRINT        ;PRINT EA BITS
2546  011620  013431                          .WORD   EABIT
2547  011622  000000                          .WORD   0
2548  011624  013737  012236  012262          MOV     EACUR,EAERR     ;GET THE EA BITS
2549  011632  006237  012262                  ASR     EAERR           ;POSITION BITS
2550  011636  006237  012262                  ASR     EAERR           ;  TO BITS 1:0
2551  011642  006237  012262                  ASR     EAERR
2552  011646  004537  005372                  JSR     R5,PRINT        ;PRINT THE OCTAL EA BITS
2553  011652  012262                          .WORD   EAERR
2554  011654  100001                          .WORD   100001
2555  011656  023727  012256  000003  90$:    CMP     ERRID,#3                ;IS IT ERROR NO. 3?
2556  011664  001010                          BNE     93$             ;BR IF NO
2557  011666  004537  005372                  JSR     R5,PRINT        ;PRINT DATA ADDRESS ASCII
2558  011672  013452                          .WORD   DATAA
2559  011674  000000                          .WORD   0
2560  011676  004537  005372                  JSR     R5,PRINT        ;PRINT DATA ADDR OCT
2561  011702  012240                          .WORD   DATAD
2562  011704  100001                          .WORD   100001
2563  011706  023727  012256  000004  93$:    CMP     ERRID,#4        ;IS IT ERROR #4?
2564  011714  001004                          BNE     94$             ;BR IF NO
2565  011716  113737  012250  012300          MOVB    DEV,EXP         ;GET DEV #
2566  011724  000416                          BR      96$
2567  011726  023727  012256  000006  94$:    CMP     ERRID,#6        ;IS IT ERROR #6?
2568  011734  001035                          BNE     95$             ;BR IF NO
2569  011736  122737  000004  012252          CMPB    #4,CMD          ;WAS IT A SENSE CMD?
2570  011744  001003                          BNE     98$             ;BR IF NO
2571  011746  005037  012300                  CLR     EXP             ;EXPECTED STATUS IS ZERO FOR SENSE
2572  011752  000403                          BR      96$
```

```
2573  011754  113737  012252  012300  98$:    MOVB    CMD,EXP          ;GET EXPECTED CMD
2574  011762  117737  000074  012276  96$:    MOVB    @DXOSA,ACTU      ;GET ACTUAL STATUS
2575  011770  004537  005372                   JSR     R5,PRINT         ;PRINT ACTUAL MESSAGE
2576  011774  013575                           .WORD   ACTUMS
2577  011776  000000                           .WORD   0
2578  012000  004537  005372                   JSR     R5,PRINT         ;PRINT STATUS OCTAL
2579  012004  012276                           .WORD   ACTU
2580  012006  100001                           .WORD   100001
2581  012010  004537  005372                   JSR     R5,PRINT         ;PRINT EXPECTED STATUS
2582  012014  013607                           .WORD   EXPMS
2583  012016  000001                           .WORD   1
2584  012020  004537  005372                   JSR     R5,PRINT         ;PRINT EXPECTED OCTAL
2585  012024  012300                           .WORD   EXP
2586  012026  100001                           .WORD   100001
2587  012030  004537  005372          95$:     JSR     R5,PRINT         ;SPACE THE PAPER
2588  012034  013625                           .WORD   CRLF1
2589  012036  000000                           .WORD   0
2590  012040  022737  001566  000042  97$:     CMP     #$ENDAD,@#42     ;UNDER ACT?
2591  012046  001001                           BNE     99$              ;BR IF NO
2592  012050  000000                           HALT                     ;HALT ON ACT ERROR
2593
2594  012052  000205                  99$:     RTS     R5               ;RETURN FROM MODULE
```

```
2595                                    ;
2596                                    ;
2597                                    ;       DX ACCESS TABLES
2598                                    ;
2599                                    ;
2600  012054  000000                    DXDSA:  .WORD   0       ;DEVICE STATUS REG ADDR.
2601  012056  000000                    DXCAA:  .WORD   0       ;COMMAND AND ADDR REG ADDR.
2602  012060  000000                    DXCSA:  .WORD   0       ;CONTROL UNIT STATUS REG ADDR
2603  012062  000000                    DXOSA:  .WORD   0       ;OFFSET AND STATUS REG ADDR.
2604  012064  000000                    DXBAA:  .WORD   0       ;BUS ADDRESS REG ADDR
2605  012066  000000                    DXBCA:  .WORD   0       ;BYTE COUNT REG ADDR
2606  012070  000000                    DXMOA:  .WORD   0       ;MAINTENANCE OUT REG ADDR.
2607  012072  000000                    DXMIA:  .WORD   0       ;MAINTENANCE IN REG ADDR
2608  012074  000000                    DXCBA:  .WORD   0       ;CONTROL BITS REG ADDR
2609  012076  000000                    DXNDA:  .WORD   0       ;NPR DATA REG ADDR
2610  012100  000000                    DXESA:  .WORD   0       ;EXTRA SIGNALS REG ADDR
2611  012102  000000                    DXMOBA: .WORD   0       ;MAINTENANCE OUT BUFFERED REG ADDR
2612  012104  000000                    DXES1A: .WORD   0       ;EXTRA EXTRA SIGNALS REG ADDR.
2613                                    ;       REGS TO BE OUTPUT TO DX
2614  012106  000000                    DXDSO:  .WORD   0       ;DEVICE STATUS REG OUT
2615  012110  000000                    DXCAO:  .WORD   0       ;COMMAND AND ADDR REG OUT
2616  012112  000000                    DXCSO:  .WORD   0       ;CONTROL UNIT STATUS REG OUT
2617  012114  000000                    DXOSO:  .WORD   0       ;OFFSET AND STATUS REG OUT
2618  012116  000000                    DXBAO:  .WORD   0       ;BUS ADDR REG. OUT
2619  012120  000000                    DXMOO:  .WORD   0       ;MAINT OUT REG ADDR
2620  012122  000000                    DXBCO:  .WORD   0
2621
```

```
2622                                    ;
2623                                    ;
2624                                    ;       DEFAULT PARAMETER TABLE
2625  012124                            DPAR:
2626  012124  000001                    DTESTN: .WORD   1       ;STARTING TEST NUMBER
2627  012126  176200                    DUBA:   .WORD   176200  ;DEFAULT UNIBUS BASE ADDR
2628  012130  000300                    DVECT:  .WORD   300     ;DEFAULT VECTOR ADDR
2629  012132  000004                    DPRIOR: .WORD   4       ;DEFAULT DX PRIORITY LEVEL
2630
```

```
2631                                    ;
2632                                    ;
2633                                    ;       INPUT PARAMETER LIMITS TABLE
2634                                    ;
2635                                    ;
2636   012134                           LPAR:
2637   012134   000001                  LTESTN: 1               ;LOWEST POSSIBLE TEST NUMBER
2638   012136   000004                  HTESTN: 4               ;HIGHEST POSSIBLE TEST NUMBER
2639   012140   160000                  LUBA:   160000          ;LOWEST POSSIBLE UNIBUS ADDRESS
2640   012142   177756                  HUBA:   177756          ;HIGHEST POSSIBLE UNIBUS ADDRESS
2641   012144   000040                  LVEC:   40              ;LOWEST POSSIBLE DX VECTOR ADDRESS
2642   012146   000776                  HVEC:   776             ;HIGHEST POSSIBLE DX VECTOR ADDRESS
2643   012150   000000                  LPRIOR: 0               ;LOWEST POSSIBLE FOR LEVEL
2644   012152   000006                  HPRIOR: 6               ;HIGHEST POSSIBLE FOR LEVEL
```

```
2645                                    ;
2646                                    ;
2647                                    ;       ACTIVE PARAMETER TABLE
2648                                    ;
2649                                    ;       THIS TABLE CONTAINS CONFIGURATION PARAMETER CURRENTLY
2650                                    ;       IN USE.
2651   012154                           APAR:
2652   012154   000000                 ATESTN: .WORD   0       ;STARTING TEST NO.
2653   012156   000000                 AUBA:   .WORD   0       ;DX UNIBUS BASE ADDR.
2654   012160   000000                 AVECT:  .WORD   0       ;DX VECTOR ADDRESS
2655   012162   000000                 APRIOR: .WORD   0       ;DX PRIORITY LEVEL
2656   012164   000000                 MSIZE:  .WORD   0       ;ADDR OF LAST WORD OF MEM, NOT INCLUDING LOADERS, KT OFF
2657   012166   000000                 MMAX:   .WORD   0       ;LAST WORD OF MEM, KT OFF
2658   012170   000000                 EASIZE: .WORD   0       ;LOWEST INVALID SETTING OF PAR
2659   012172   000000                 KT:     .WORD   0       ;=0 NO KT IN USE =1 KT IN USE
2660   012174   000000                 FIRSTP: .WORD   0       ;0 IF FIRST PASS, 1 IF NOT FIRST PASS
2661   012176   177564                 TPS:    .WORD   177564  ;TTY PRINTER STATUS REG ADDR
2662   012200   177566                 TPB:    .WORD   177566  ;TTY PRINTER BUFFER REG ADDR
2663   012202   177560                 TKS:    .WORD   177560  ;TTY KEYBOARD STATUS REG ADDR
2664   012204   177562                 TDB:    .WORD   177562  ;TTY KEYBOARD BUFFER.
2665   012206   000000                 CTESTN: .WORD   0       ;CURRENT TEST NUMBER
2666   012210   000000                 PARMV:  .WORD   0       ;0 IF PARMS NOT VALID, 1 IF THEY ARE VALID
2667   012212   000001                 COUNT:  .WORD   1       ;ITERATION COUNT CONSTANT
2668   012214   000000                 CNT:    .WORD   0       ;ITERATION VARIABLE
2669   012216   000000                 ERR:    .WORD   0       ;IF ERROR DETECTED, CONTAINS TN
2670   012220   000000                 LOCK:   .WORD   0       ;SW=TN IF LOOP SW 14 ERR DETECTED
2671   012222   000000                 LOCKAD: .WORD   0       ;LOCK ON ERROR ADDR
2672   012224   000000                 EABITS: .WORD   0       ;HIGHEST VALID SETTING OF ADDR 17 16
2673   012226   000000                 PHYMAX: .WORD   0       ;LOWEST INVALID PHYSICAL ADDR, LOW 16 BITS
2674   012230   000000                 BUFBAS: .WORD   0       ;START OF CURRENT WORKING BUFFER, VIRTUAL
2675   012232   000000                 BUFSZE: .WORD   0       ;SIZE OF CURRENT BUFFER IN OCT BYTES
2676   012234   000000                 BUFMAX: .WORD   0               ;MAX VIRTUAL ADDR FOR THIS KT SETTING
2677   012236   000000                 EACUR:  .WORD   0               ;CURRENT SETTING OF EA BITS
2678   012240   000000                 DATAD:  .WORD   0       ;CURRENT DATA ADDR
2679   012242   000000                 BFBAPH: .WORD   0       ;CURRENT PHYSICAL BUFFER BASE ADDR
2680   012244   000000                 BFADPH: .WORD   0               ;CURRENT PHYSICAL DATA ADDR
2681   012246   000000                 SPWPHY: .WORD   0       ;PHYSICAL ADDR OF SPW START
2682   012250   000000                 DEV:    .WORD   0       ;CURRENT IBM DEVICE NO.
2683   012252   000400                 CMD:    .WORD   400     ;CURRENT IBM COMMAND WITH PARITY BIT
2684   012254   000000                 ERRPC:  .WORD   0       ;PC WHERE ERROR WAS DETECTED
2685   012256   000000                 ERRID:  .WORD   0               ;ERROR NUMBER
2686   012260   000000                 ERRMS:  .WORD   0               ;ADDR OF UNIQUE ERROR MESSAGE
2687   012262   000000                 EAERR:  .WORD   0               ;EABITS ON ERROR
2688   012264   000000                 RECORD: .WORD   0               ;SWITCH TO CONTROL DEV REC RECORDING
2689   012266   000000                 FRSTD:  .WORD   0               ;FIRST DEVICE RECOGNIZED SW
2690   012270   000000                 FRSTDV: .WORD   0               ;FIRST DEVICE RECOGNIZED NUMBER
2691   012272   000000                 DEVCT:  .WORD   0               ;FIRST DEVICE GROUP COUNT
2692   012274   000000                 DEVMSS: .WORD   0               ;DEVICE MESSAGE PRINTED SW
2693   012276   000000                 ACTU:   .WORD   0       ;ACTUAL DATA
2694   012300   000000                 EXP:    .WORD   0               ;EXPECTED DATA
2695   012302   000000                 DEVFND: .WORD   0               ;= 1 IF DEVICE EVER RECOGNIZED
2696   012304   000000                 SPWMVD: .WORD   0       ;= 1 IF SPW TAPLE IN ALTERNATE LOCATION
```

```
2697                            .NLIST  BIN
2698  012306  ERR1:    .ASCIZ  /FAST NPR DATA ERROR/
2699  012314
2700  012322
2701  012330
2702  012332  ERR2:    .ASCIZ  /NO DX INTERRUPT/
2703  012340
2704  012346
2705
2706  012352  ERR3:    .ASCIZ  /DX ADDRESSING ERROR, DATA XFER/
2707  012360
2708  012366
2709  012374
2710  012402
2711  012410
2712  012411  ERR4:    .ASCIZ  /SPW ACCESS, ADDR ERROR/
2713  012416
2714  012424
2715  012432
2716  012440  ERR5:    .ASCIZ  /NO IBM DEV ADR RECOGNIZED. CHECK A20 MODULE/
2717  012446
2718  012454
2719  012462
2720  012470
2721  012476
2722  012504
2723  012512
2724  012514  ERR6:    .ASCIZ  /DST ACCESS ERROR/
2725  012522
2726  012530
2727  012535  T4MSG:   .ASCIZ  /TST3 MUST BE RUN FIRST FOR TST 4 TO WORK/
2728  012542
2729  012550
2730  012556
2731  012564
2732  012572
2733  012600
2734  012606  HEADER: .ASCII  'MD-11-DZDXJC  DX11B MEMORY ADDRESSING TEST'<15><12>
2735  012614
2736  012622
2737  012630
2738  012636
2739  012644
2740  012652
2741  012660
2742  012662          .ASCII  'TYPE:  <D>, DEFAULT PARAMETERS'<15><12>
2743  012670
2744  012676
2745  012704
2746  012712
2747  012720
2748  012721          .ASCII  '       <P>, PREVIOUS PARAMETERS'<15><12>
2749  012726
2750  012734
2751  012742
2752  012750
```

```
2753  012754          .ASCII  '       <S>, SELECT PARAMETERS'<15><12>
2754  012762
2755  012770
2756  012776
2757  013004
2758  013005          .ASCIZ  '       <N>, START AT THIS TEST NO.'<15><12>
2759  013012
2760  013020
2761  013026
2762  013034
2763  013042
2764  013044  INVM:   .ASCIZ  'INVALID ENTRY'
2765  013052
2766  013060
2767  013062  INVPRM: .ASCIZ  'NEED MORE PARMS'
2768  013070
2769  013076
2770  013102  PROMPT: .ASCIZ  'D,P,S,N? '
2771  013110
2772  013114  TESTN:  .ASCIZ  'STARTING TEST NO. '
2773  013122
2774  013130
2775  013136
2776  013137  BASEA:  .ASCIZ  'BASE ADDRESS '
2777  013144
2778  013152
2779  013155  VECTA:  .ASCIZ  'VECTOR ADDRESS '
2780  013162
2781  013170
2782  013175  PRILV:  .ASCIZ  'DX PRIORITY LEVEL '
2783  013202
2784  013210
2785  013216
2786  013220  SETSW:  .ASCIZ  'SET SWITCHES '
2787  013226
2788  013234
2789  013236  BELL:   .ASCIZ  <207> 'END PASS'
2790  013244
2791  013250  BADP:   .ASCIZ  'INPUT PARAMETER OUTSIDE LIMITS'
2792  013256
2793  013264
2794  013272
2795  013300
2796  013306
2797  013307  ERPC:   .ASCIZ  'ERROR PC: '
2798  013314
2799  013322  ERNO:   .ASCIZ  '  ERR NO.: '
2800  013330
2801  013336  ERTN:   .ASCIZ  'TEST NO.: '
2802  013344
2803  013351  ERADR:  .ASCIZ  'VIRT ADDR OF BUFFER BASE: '
2804  013356
2805  013364
2806  013372
2807  013400
2808  013404  ERADPH: .ASCIZ  'PHY ADDR OF BUFFER: '
```

```
2809  013412
2810  013420
2811  013426
2812  013431  EABIT:  .ASCIZ  'EA BITS IN OCT: '
2813  013436
2814  013444
2815  013452  DATAA:  .ASCIZ  'VIRT ADDR OF ACCESS ERROR: '
2816  013460
2817  013466
2818  013474
2819  013502
2820  013506  DEVMS:  .ASCIZ  '1ST IBM DEV RCGNZD.  OCTAL:  '
2821  013514
2822  013522
2823  013530
2824  013536
2825  013544  DEVCMS: .ASCIZ  'SIZE OF 1ST DEV GROUP:  '
2826  013552
2827  013560
2828  013566
2829  013574
2830  013575  ACTUMS: .ASCIZ  'ACTUAL:  '
2831  013602
2832  013607  EXPMS:  .ASCIZ  '  EXPECTED:  '
2833  013614
2834  013622
2835  013625  CRLF1:  .ASCIZ  / /<15><12>
2836                          .LIST   BIN
2837          013632          .EVEN
2838                          ;INBUF MUST BE ON AN EVEN WORD BOUNDARY
2839  013632      012 INBUF:  .BYTE   12      ;KEYBOARD INPUT BUFFER
2840          014000          .=14000
2841  014000          SPW:                    ;START OF STATIONARY SPW TABLE USED ONLY TO
2842                                          ;FIELD UNEXPECTED ERRORS.
2843  014000  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2844  014002  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2845  014004  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2846  014006  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2847  014010  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2848  014012  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2849  014014  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2850  014016  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2851  014020  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2852  014022  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2853  014024  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2854  014026  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2855  014030  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2856  014032  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2857  014034  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2858  014036  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2859  014040  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2860  014042  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2861  014044  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2862  014046  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2863  014050  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2864  014052  000002          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
```

```
2865  014054  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2866  014056  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2867  014060  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2868  014062  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2869  014064  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2870  014066  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2871  014070  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2872  014072  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2873  014074  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2874  014076  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2875  014100  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2876  014102  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2877  014104  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2878  014106  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2879  014110  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2880  014112  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2881  014114  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2882  014116  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2883  014120  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2884  014122  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2885  014124  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2886  014126  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2887  014130  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2888  014132  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2889  014134  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2890  014136  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2891  014140  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2892  014142  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2893  014144  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2894  014146  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2895  014150  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2896  014152  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2897  014154  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2898  014156  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2899  014160  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2900  014162  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2901  014164  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2902  014166  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2903  014170  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2904  014172  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2905  014174  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2906  014176  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2907  014200  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2908  014202  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2909  014204  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2910  014206  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2911  014210  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2912  014212  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2913  014214  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2914  014216  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2915  014220  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2916  014222  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2917  014224  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2918  014226  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2919  014230  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2920  014232  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
```

```
2921  014234  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2922  014236  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2923  014240  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2924  014242  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2925  014244  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2926  014246  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2927  014250  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2928  014252  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2929  014254  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2930  014256  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2931  014260  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2932  014262  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2933  014264  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2934  014266  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2935  014270  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2936  014272  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2937  014274  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2938  014276  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2939  014300  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2940  014302  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2941  014304  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2942  014306  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2943  014310  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2944  014312  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2945  014314  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2946  014316  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2947  014320  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2948  014322  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2949  014324  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2950  014326  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2951  014330  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2952  014332  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2953  014334  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2954  014336  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2955  014340  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2956  014342  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2957  014344  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2958  014346  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2959  014350  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2960  014352  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2961  014354  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2962  014356  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2963  014360  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2964  014362  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2965  014364  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2966  014366  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2967  014370  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2968  014372  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2969  014374  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2970  014376  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2971  014400  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2972  014402  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2973  014404  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2974  014406  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2975  014410  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2976  014412  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
```

```
2977  014414  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2978  014416  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2979  014420  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2980  014422  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2981  014424  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2982  014426  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2983  014430  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2984  014432  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2985  014434  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2986  014436  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2987  014440  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2988  014442  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2989  014444  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2990  014446  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2991  014450  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2992  014452  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2993  014454  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2994  014456  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2995  014460  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2996  014462  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2997  014464  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2998  014466  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2999  014470  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3000  014472  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3001  014474  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3002  014476  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3003  014500  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3004  014502  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3005  014504  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3006  014506  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3007  014510  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3008  014512  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3009  014514  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3010  014516  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3011  014520  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3012  014522  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3013  014524  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3014  014526  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3015  014530  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3016  014532  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3017  014534  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3018  014536  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3019  014540  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3020  014542  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3021  014544  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3022  014546  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3023  014550  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3024  014552  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3025  014554  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3026  014556  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3027  014560  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3028  014562  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3029  014564  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3030  014566  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3031  014570  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3032  014572  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
```

```
3033  014574  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3034  014576  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3035  014600  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3036  014602  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3037  014604  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3038  014606  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3039  014610  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3040  014612  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3041  014614  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3042  014616  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3043  014620  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3044  014622  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3045  014624  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3046  014626  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3047  014630  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3048  014632  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3049  014634  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3050  014636  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3051  014640  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3052  014642  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3053  014644  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3054  014646  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3055  014650  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3056  014652  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3057  014654  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3058  014656  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3059  014660  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3060  014662  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3061  014664  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3062  014666  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3063  014670  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3064  014672  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3065  014674  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3066  014676  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3067  014700  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3068  014702  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3069  014704  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3070  014706  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3071  014710  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3072  014712  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3073  014714  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3074  014716  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3075  014720  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3076  014722  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3077  014724  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3078  014726  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3079  014730  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3080  014732  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3081  014734  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3082  014736  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3083  014740  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3084  014742  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3085  014744  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3086  014746  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3087  014750  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
3088  014752  000002                .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
```

```
 3089   014754   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3090   014756   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3091   014760   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3092   014762   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3093   014764   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3094   014766   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3095   014770   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3096   014772   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3097   014774   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3098   014776   000002                          .WORD   2       ;STATUS OF UNIT CHECK WITH NO DST ACCESS
 3099   015000                           TT:                     ;START OF STATIONARY TUMBLE TABLE
 3100   016000                           ENDALL:
 3101            000001                          .END
```

```
ACTU    012276          2574*   2579    2693#
ACTUMS  013575          2576    2830#
ADRO  = 004000           616#   2156    2163    2316    2318
APAR    012154          2651#
APRIOR  012162           822*    840*    878*   2655#
ATESTN  012154           736     819*    837*    857*    893*   2652#
AUBA    012156           820*    838*    864*    909    2653#
AVECT   012160           821*    839*    871*   1907    2050    2654#
BADP    013250          1573    2791#
BASEA   013137           971    2776#
BELL    013236           782    2789#
BFADPH  012244          2056*   2060    2081*   2680#
BFBAPH  012242          1915    2056    2150    2307    2345    2382*   2404*   2405*   2426    2434    2443    2450    2543
                        2679#
BIT0  = 000001           560#
BIT00 = 000001           550#    560
BIT01 = 000002           549#    559
BIT02 = 000004           548#    558
BIT03 = 000010           547#    557
BIT04 = 000020           546#    556
BIT05 = 000040           545#    555
BIT06 = 000100           544#    554
BIT07 = 000200           543#    553
BIT08 = 000400           542#    552
BIT09 = 001000           541#    551
BIT1  = 000002           559#
BIT10 = 002000           540#
BIT11 = 004000           539#
BIT12 = 010000           538#
BIT13 = 020000           537#
BIT14 = 040000           536#
BIT15 = 100000           535#
BIT2  = 000004           558#
BIT3  = 000010           557#
BIT4  = 000020           556#
BIT5  = 000040           555#
BIT6  = 000100           554#
BIT7  = 000200           553#
BIT8  = 000400           552#
BIT9  = 001000           551#
BPTVEC= 000014           567#
BUFAD   010620          1904    1945    2048    2087    2139    2183    2282    2341    2370#
BUFBAS  012230          1927    1981    2055    2144    2287    2381*   2391*   2392*   2394    2402*   2403*   2412    2419*
                        2420*   2421    2455    2535    2674#
BUFCLR  007030          1912    1944    1981#   2054    2086    2142    2182    2285    2340
BUFMAX  012234          2083    2396*   2425*   2455*   2456*   2457    2676#
BUFSZE  012232          1910*   1928    1982    2053*   2141*   2284*   2675#
CHECKP  005224           858     865     872     879     894    1542#
CKSWR = 104406          1838#
CMD     012252          2138*   2165    2167    2310*   2312    2320    2322    2324    2329    2337*   2338    2569    2573
                        2683#
CMDO  = 002000           614#   1963    1964    2166    2168    2321    2323
CNT     012214           735*    759*   2668#
COUNT   012212           735    2667#
CR    = 000015           475#    952    1697    1707
CRLF  = 000200           476#   1649    1707
```

```
CRLF1   013625          1604    2588    2835#
CTESTN  012206           736*    739     756*    757     2482    2495    2527    2665#
DATAA   013452          2558    2815#
DATAD   012240          2055*   2071    2077*   2082*   2083    2561    2678#
DDISP = 177570           482#    667     720
DEV     012250          2143*   2145    2146*   2147    2149*   2153    2154    2155    2164    2169    2179*   2180    2233
                        2278*   2279*   2281    2293    2314    2315    2319    2565    2682#
DEVCMS  013544           772    2825#
DEVCT   012272           763     775    2136*   2187    2235*   2241    2691#
DEVFND  012302          2234*   2272    2695#
DEVMS   013506           766    2820#
DEVMSS  012274           733*    761     777*   2692#
DISPLA  001142           667#    720*    728*
DISPRE  000174           624#    728
DONE  = 000200           610#   1936    2013    2078
DONTSV  010140          2177    2241#
DPAR    012124          2625#
DPRIOR  012132           822     840     999    2629#
DSWR  = 177570           481#    666     719
DTESTN  012124           819     837     966    2626#
DUBA    012126           820     838     977    2627#
DVECT   012130           821     839     988    2628#
DXBAA   012064          1915*   2060*   2604#
DXBAO   012116          2618#
DXBCA   012066          1954*   2605#
DXBCO   012122          2620#
DXCAA   012056          2154*   2314*   2601#
DXCAO   012110          2615#
DXCBA   012074          2022    2157    2159    2608#
DXCSA   012060          1914*   1936*   1938*   1959*   2013*   2014*   2015*   2016*   2057*   2078*   2080*   2151*   2313*
                        2602#
DXCSO   012112          2616#
DXDSA   012054           908    2600#
DXDSO   012106          2614#
DXESA   012100          1937*   1965*   2012*   2079*   2610#
DXES1A  012104          2612#
DXGO    006706          1916    1953#   2061
DXMIA   012072          2607#
DXMOA   012070          1953*   1961*   1962*   1963*   1964*   2155*   2156*   2162*   2163*   2164*   2165*   2166*   2167*
                        2168*   2174*   2175*   2315*   2316*   2317*   2318*   2319*   2320*   2321*   2322*   2323*   2334*
                        2335*   2606#
DXMOBA  012102          2611#
DXMOO   012120          1913*   1929    1939    1941*   1953    2058*   2619#
DXNDA   012076          2609#
DXOSA   012062          2018*   2021*   2150*   2169    2296    2326    2329    2446*   2452*   2574    2603#
DXOSO   012114          2617#
DXRES   007100          1903    2012#   2025    2047    2137    2178    2271    2336
EABIT   013431          2546    2812#
EABITS  012224          1378*   1399*   1400*   1401*   1402*   2408    2428    2672#
EACUR   012236          1914    2057    2151    2301    2313    2374*   2407*   2408    2428    2439    2548    2677#
EAERR   012262          2548*   2549*   2550*   2551*   2553    2687#
EASIZE  012170          1377*   1389*   1390*   2415    2658#
EMTVEC= 000030           570#
ENDALL  016000          2376    3100#
ENDB    010734          2385    2394#
ENDB1   011270          2397    2418    2454    2458    2460#
```

```
ENDPRM  002270              816     916#
ENDTST  001410              748#   1949    2091    2194    2349
ENDT1   006670             1945#
END1    001576              785     791#
END4    010614             2277    2344    2349#
EOP     001452              761#
ERADPH  013404             2540    2808#
ERADR   013351             2532    2803#
ERNO    013322             2512    2799#
ERPC    013307             2505    2797#
ERR     012216              738*   2004*   2482*   2669#
ERRID   012256             2499*   2515    2529    2555    2563    2567    2685#
ERRMS   012260             2500*   2517    2686#
ERRPC   012254             2498*   2507*   2509    2684#
ERRR    011272             1931    1971    2074    2171    2190    2331    2478#
ERRVEC= 000004              563#    717     718*    729*   1430    1431    1435*  1440*  1443*  1469*  1474*  1478   1495*
                           1505*   1506*
ERR1    012306             1933    2698#
ERR2    012332             1973    2702#
ERR3    012352             2076    2706#
ERR4    012411             2173    2712#
ERR5    012440             2193    2716#
ERR6    012514             2333    2724#
ERTN    013336             2524    2801#
EXP     012300             2565*   2571*   2573*   2585    2694#
EXPMS   013607             2582    2832#
FIRSTP  012174              732*    814     915*   2660#
FRSTD   012266             2135*   2230    2232*   2689#
FRSTDV  012270              769    2233*   2279    2690#
GETPRM  001600              734     814#
GNS   = ****** U            623    1831    1832    1833    1834    1836    1838    1839    1840    1841
GO      001346              736#    760
GO1     006526             1914#   1942
GTSWR = 104405             1836#
HEADER  012606              830    2734#
HLDO  = 040000              613#   1961    1962    2162    2317
HPRIOR  012152             1568    2644#
HT    = 000011              473#   1647    1707
HTESTN  012136              757    1547    2638#
HUBA    012142             1554    2640#
HVEC    012146             1561    2642#
INASC   002532              886     948     962     973     984     995    1020#
INBUF   013632             1025*   1031*   1032    2839#
INIT    007050             1902    1999#   2046    2133    2270
INPRM   002272              832     854     862     869     876     891     941#
INTERR  007026             1921    1967*   1974*   1976#   2006*   2065
INTR1   006550             1908    1921#
INTR2   007306             2051    2065#
INVM    013044              848    2764#
INVPRM  013062              901    2767#
IOTVEC= 000020              568#
KIPAR0= 172340              601#   1411    1461
KIPAR1= 172342              602#
KIPAR2= 172344              603#
KIPAR3= 172346              604#
KIPAR4= 172350              605#
```

```
KIPAR5= 172352            606#   2304*
KIPAR6= 172354            607#   2387*   2390*   2414*   2415
KIPAR7= 172356            608#
KIPDR0= 172300            590#
KIPDR1= 172302            591#
KIPDR2= 172304            592#
KIPDR3= 172306            593#
KIPDR4= 172310            594#
KIPDR5= 172312            595#
KIPDR6= 172314            596#
KIPDR7= 172316            597#
KT      012172            778    1366*   1385*   2299    2384    2410    2537    2659#
LF    = 000012            474#   1701    1707
LOCK    012220            737*   2487    2495*   2497*   2670#
LOCKAD  012222           2005*   2491    2671#
LOOP1   006506           1910#   1948
LOOP2   007230           2053#   2090
LOOP3   007516           2143#   2186
LOOP4   010250           2286#   2348
LOP2    007256           2057#   2084
LOP3    007552           2150#   2181
LOP4    010364           2311#   2339
LPAR    012134           2636#
LPRIOR  012150           1566    2643#
LTESTN  012134           1545    2637#
LUBA    012140           1552    2639#
LVEC    012144           1559    2641#
MMAX    012166           1367*   2443    2657#
MMVEC = 000250            579#
MSIZE   012164           1368*   1371*   1376*   2441    2457    2656#
NEXTB   010750           2373    2402#   2424    2436    2445
PARMV   012210            826     844     853*    898     914*   2666#
PHYMAX  012226           1391*   1392*   1393*   1394*   1395*   1396*   1397*   1398*   2430    2673#
PIRQ  = 177772            480#
PIRQVE= 000240            574#
PRILV   013175            993    2782#
PRINT   005372            765     768     771     774     781     829     847     883     900     945     959     970     981
                          992    1572    1596#   2274    2504    2508    2511    2514    2518    2523    2526    2531    2534
                         2539    2542    2545    2552    2557    2560    2575    2578    2581    2584    2587
PROMPT  013102            946    2770#
PR0   = 000000            497#
PR1   = 000040            498#
PR2   = 000100            499#
PR3   = 000140            500#
PR4   = 000200            501#
PR5   = 000240            502#
PR6   = 000300            503#
PR7   = 000340            504#
PS    = 177776            477#    478
PSW   = 177776            478#
PTYGEN  010034           2152    2211#   2280    2311
PTY2    010042           2214#   2218
PTY3    010050           2215    2217#
PTY4    010066           2220    2223#
PTY5    010072           2222    2224#
PWRVEC= 000024            569#    713*    714*   1847*   1848*   1857*   1863*   1875*   1876*
```

```
RDCHR = 104407          1242    1839#
RDLIN = 104410          1024    1317    1840#
RDOCT = 104411          1030    1841#
RECORD  012264          2134*   2228    2243*   2688#
RESTAR  001334           734#    791    1880
RESVEC= 000010           564#
SAVDEV  010074          2161    2228#
SELO  = 020000           612#   1961    1962    2162    2317
SETSW   013220           884    2786#
SIZE    004224           906    1364#
SOSIEN= 000004           611#   1937    1965    2079
SPW     014000          2018    2021    2446    2448    2450    2452    2841#
SPWMVD  012304          2019    2447*   2453*   2696#
SPWPHY  012246          2681#
SRVO  = 001000           615#   2174    2175    2334    2335
SR0   = 177572           583#    780*   1441    1473*   1484*   2393*
SR1   = 177574           584#
SR2   = 177576           585#
SR3   = 172516           586#   1470*   1487*
STACK = 001100           468#    709
START   001164           627     702#   1083    1166
STKLMT= 177774           479#
SWR     001140           666#    707     719*    721     727*    754    1086    1125    1180*   1379    1855    1868*   2478
                        2489    2493    2501    2521
SWREG   000176           625#    727    1086    1125    1148
SW0   = 000001           532#
SW00  = 000001           522#    532
SW01  = 000002           521#    531
SW02  = 000004           520#    530
SW03  = 000010           519#    529
SW04  = 000020           518#    528
SW05  = 000040           517#    527
SW06  = 000100           516#    526
SW07  = 000200           515#    525
SW08  = 000400           514#    524
SW09  = 001000           513#    523
SW1   = 000002           531#
SW10  = 002000           512#   1379
SW11  = 004000           511#
SW12  = 010000           510#   2521
SW13  = 020000           509#   2501
SW14  = 040000           508#   2489    2493
SW15  = 100000           507#   2478
SW2   = 000004           530#
SW3   = 000010           529#
SW4   = 000020           528#
SW5   = 000040           527#
SW6   = 000100           526#
SW7   = 000200           525#
SW8   = 000400           524#
SW9   = 001000           523#
TBITVE= 000014           565#
TDB     012204          2664#
TESTN   013114           960    2772#
TKS     012202          2663#
TKVEC = 000060           572#   1058*   1059*
```

```
TPB     012200            2662#
TPS     012176            2661#
TPVEC = 000064             573#
TRAPVE= 000034             571#    711*    712*
TRTVEC= 000014             566#
TST     001364             739#    755     758
TSTAB   001376             741     742#
TST1    006454             743    1902#
TST2    007176             744    2046#
TST3    007444             745    2133#
TST4    010156             746    2270#
TT      015000            3099#
TYPE  = 104401            1080    1092    1146    1147    1150    1161    1171    1182    1201    1249    1255    1260    1264
                          1269    1270    1272    1275    1279    1346    1348    1603    1611    1652    1768    1831#   1877
TYPOC = 104402            1149    1608    1832#
TYPON = 104404            1834#
TYPOS = 104403            1833#
T4MSG   012535            2275    2727#
VECTA   013155             982    2779#
$AUTOB  001134             663#    1139    1297
$BDADR  001122             658#
$BDDAT  001126             660#
$BELL   004016            1092    1289#
$CHARC  005742            1654*   1664*   1671    1699*   1704#
$CKSWR  003120            1125#   1838
$CMTAG  001100             646#    704     705
$CM3  = 000000             676#
$CNTLC  004022            1080    1161    1290#
$CNTLG  004034            1146    1292#
$CNTLU  004027            1171    1264    1291#
$CORE   005112            1439    1495#
$CRLF   001161             677#    1182    1269    1289    1351    1653    1707
$CROUT  005142            1495    1502#
$ENDAD  001566             634     787#    824     2590
$ERFLG  001103             649#
$ERMAX  001115             655#
$ERRPC  001116             656#
$ERRTB  001164             694#
$ERTTL  001112             653#
$FILLC  001156             674#    1657    1707
$FILLS  001155             673#    1707
$GDADR  001120             657#
$GDDAT  001124             659#
$GTSWR  003210            1147#   1836
$HD   = 000003             463     464
$HIOCT  004222            1339*   1350#
$ICNT   001104             650#
$ILLUP  006436            1847    1863    1882#
$INTAG  001135             664#    1144*   1163    1183    1297
$ITEMB  001114             654#
$KTNEX  005104            1440    1494#
$KTOUT  005042            1474    1483#   1492
$KT11   004574            1364*   1381*   1383    1438#   1442*   1494*
$LF     001162             678#    1279    1289    1351    1707
$LPADR  001106             651#
$LPERR  001110             652#
```

```
$LSTAD  005220          1367    1368    1369    1509*   1517#
$LSTBK  005222          1386    1388*   1389    1391    1399    1510*   1518#
$MAIL = ****** U         731    1641
$MAP    004666          1446*   1450*   1452#   1475    1485
$MAPRG  004672          1447    1454#
$MMOUT  005064          1477    1489#
$MNEW   004052          1150    1295#
$MSWR   004041          1147    1293#
$NOMAP  004712          1451    1459#
$NULL   001154           672#   1659    1707
$OCNT   006170          1740*   1769*   1782#
$OMODE  006172          1735*   1739*   1744    1747*   1758*   1784#
$PASS   001100           647#
$POWER  006444          1878    1885#
$PWRAD  006432          1880#
$PWRDN  006272           713    1847#   1875
$PWRMG  006426          1878#
$PWRUP  006344          1857    1863#
$QUES   001160           676#   1201    1272    1289    1348    1351    1707
$RDCHR  003462          1214#   1839
$RDDEC= ****** U        1842
$RDLIN  003552          1237#   1840
$RDOCT  004064          1312#   1841
$RDSZ = 000010          1230#
$R2A  = ****** U        1842
$SAVRE= ****** U        1842
$SAVR6  006442          1856*   1864    1865*   1866*   1884#
$SETUP= 000114           702#    710     711     713     715    1084    1089    1090    1120    1297
$SIZE   004510          1365    1382    1421#
$SIZEX  005146          1486    1488    1503#
$STOP   004670          1445*   1448*   1453#   1481
$STUP = 177777           702#
$SVPC = 000204           632#    637
$SWR  = 160000           463     464#    676     1881
$TKB    001146           669#   1040    1060    1071    1101    1129    1156    1678    1685    1707
$TKCNT  002602          1041#   1055*   1090    1107*   1221    1223*
$TKINT  002620          1020    1055#   1081    1142
$TKQEN= 002620          1045#   1115    1226
$TKQIN  002604          1042#   1056*   1057    1113*   1114*   1115    1117*
$TKQOU  002606          1043#   1057*   1224    1225*   1226    1228*
$TKQSR  002610          1044#   1056    1117    1228
$TKS    001144           668#   1040    1061*   1097*   1099    1105*   1127    1143*   1153    1165*   1185*   1676    1683
                        1707
$TKSRV  002670          1058    1071#
$TN   = 000001           463#
$TPB    001152           671#   1696*   1707
$TPFLG  001157           675#   1635    1707
$TPS    001150           670#   1692    1707
$TRAP   006174           711    1804#
$TRAP2  006234          1819#   1830
$TRP  = 000012          1823#   1832#   1833#   1834#   1835#   1836    1837#   1838    1839#   1840#   1841#   1842#
$TRPAD  006246          1813    1830#
$TSTNM  001102           648#
$TTYIN  004006          1239    1240    1252    1270    1284    1288#
$TYPBN= ****** U        1835
$TYPDS= ****** U        1835
```

```
$TYPE   005444          1635#   1823    1831
$TYPEC  005614          1187    1656    1663    1670    1675#
$TYPEX  005744          1695    1700    1702    1705#
$TYPOC  005772          1738#   1832
$TYPON  006006          1737    1740#   1834
$TYPOS  005746          1733#   1833
$XOFF = 000023          1680    1707
$XON  = 000021          1073    1687    1694    1707
$OFILL  006171          1734*   1738*   1748    1783#
.     = 016000           619#    623#    632     633#    635#    637#    638#    645#    679     708     1040    1044#   1045
                        1046    1288#   1289    1290    1297#   1351    1707    1859    1883    2837#   2840#
```

```
CKSWR     453#
COMMEN      1#    575#
ENDCOM      1#    575#
ERROR     469#
ESCAPE      1#    575#
GETPRI      1#    575#   1434
GETSWR      1#    453#    575#
MULT        1#    575#
NEWTST      1#    575#
POP         1#    453#    575#   1031    1340    1613    1868    1869
PUSH        1#    453#    575#   1314    1596    1849    1855
REPORT      1#    575#
SCOPE     470#
SETPRI      1#    453#    575#    748    1217    1955    1999    2483
SETTRA   1823#   1832    1833    1834    1836    1838    1839    1840    1841
SETUP       1#    575#    702
SKIP        1#    575#
SLASH       1#    575#
SPACE     575#
STARS       1#    575#    630     641     679    1039    1120    1135    1206    1230    1300    1407    1620    1710    1798
         1845    1861
SWRSU       1#    575#    715#
TRMTRP   1823#
TYPBIN      1#    575#
TYPDEC      1#    575#
TYPNAM      1#    575#
TYPNUM      1#    575#
TYPOCS      1#    575#
TYPOCT      1#    575#   1148
TYPTXT      1#    575#
$$CMRE    639#
$$CMTM    639#
$$ESCA      1#    575#
$$NEWT      1#    575#
$$SET    1823#   1832    1833    1834    1836    1838    1839    1840    1841
$$SKIP      1#    575#
.EQUAT      1#    453#    465
.HEADE      1#    453#
.KT11       1#    453#    575
.SETUP      1#    453#    702
.SWRHI      1#
.$ACT1      1#    453#    628
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#    453#    617
.$CMTA      1#    453#    639
.$DB2D      1#
.$DB2O      1#
.$DIV       1#
.$EOP       1#
.$ERRO      1#
.$ERRT      1#
.$MULT      1#
.$POWE      1#    453#   1843
```

```
.$RAND      1#
.$RDDE      1#
.$RDOC      1#    453#   1298
.$READ      1#    453#   1037
.$R2AZ      1#
.$SAVE      1#
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#
.$SIZE      1#    453#   1405
.$SUPR      1#
.$TRAP      1#    453#   1796
.$TYPB      1#
.$TYPD      1#
.$TYPE      1#    453#   1618
.$TYPO      1#    453#   1708
.$40CA      1#
.1170       1#


. ABS.  016000     000

 ERRORS DETECTED:  0

 CZDXJC,CZDXJC/SOL/CRF/NL:TOC=SYSMAC.C5,CZDXJC.P11
 RUN-TIME: 14 12 .8 SECONDS
 RUN-TIME RATIO: 54/28=1.9
 CORE USED:  33K  (65 PAGES)
```