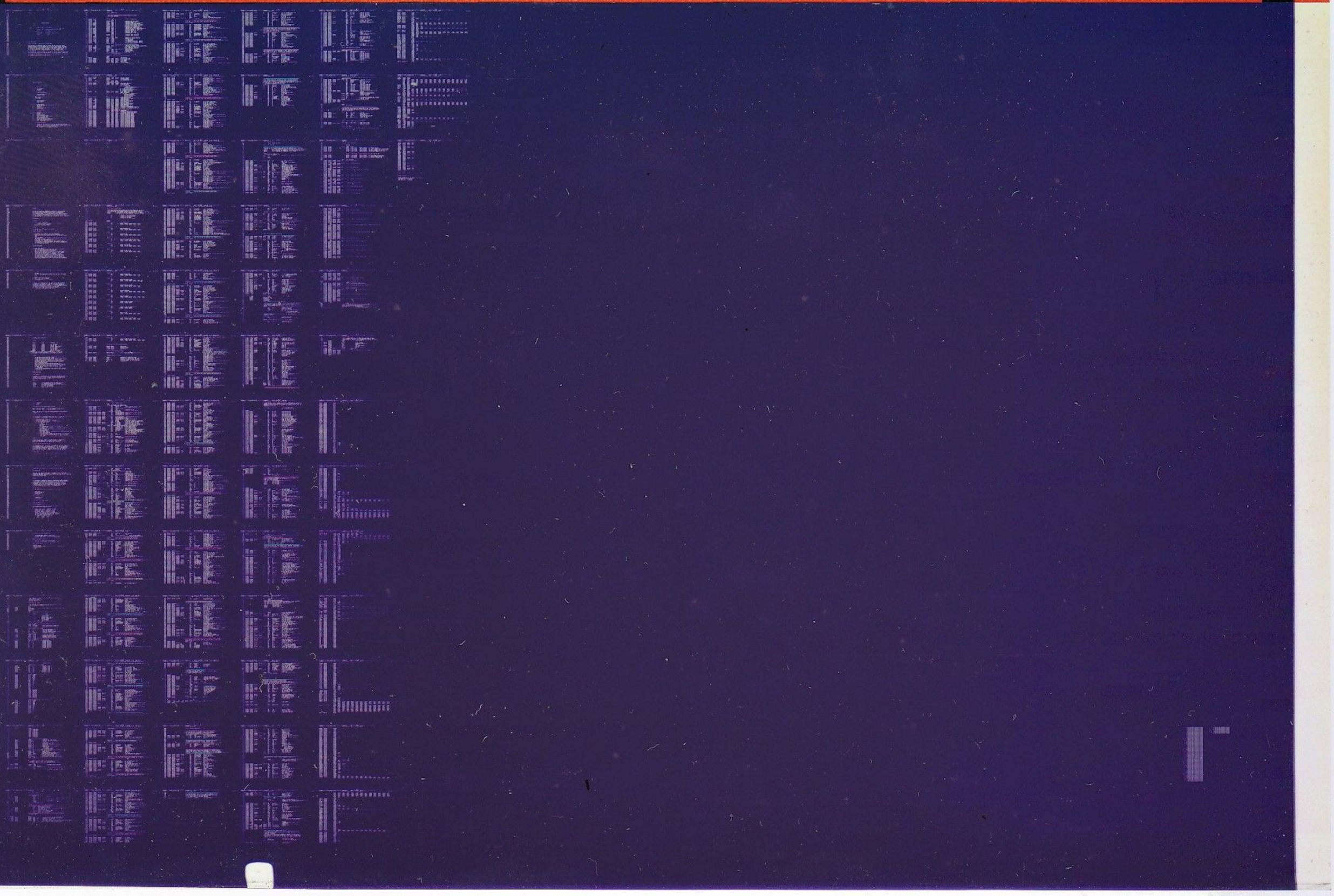
DRV11-B

DMA INTFC DIAG
CNDRAAO

AH-T450A-MC
FICHE 1 OF 1



```
                                .REM %

                                        IDENTIFICATION
                                        --------------

                                PRODUCT CODE:   AC-T449A-MC
                                PRODUCT NAME:   CNDRAA0 DRV11B DMA INTFC DIAG
                                DATE:           DEC 1982
                                MAINTAINER:     DIAGNOSTIC SERVICES/ISS
                                AUTHOR:         T.CHAPSKY
```

TABLE OF CONTENTS

CVDRAC WAS MADE SPECIFIC TO 11/21 PROCESSOR BY CHANGING
PRIORITY 7 TO 6 AND ALSO CHANGING DEFAULT ADDRESS AND VECTOR
AND ALSO ADDING A CALL TO CNMAC2.SML TO INIT ODT AND LTC VECTORS.
THE DIAGNOSTIC WAS RENAMMED TO CNDRAA0.

5779 11.0 LISTING

```
                                1.0     ABSTRACT
                                        --------

                                        THE DRV11B DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED
                                        TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE WITH
                                        THE LOOP BACK CABLE INSERTED IN THE USER I/O CONNECTORS.
                                        TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL
                                        VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5
                                        OF THIS DOCUMENT. IF THE SYSTEM ALSO INCLUDES AN "REV11" (DMA REFRESH),
                                        THE DMA REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.

                                2.0     REQUIREMENTS
                                        ------------

                                2.1     EQUIPMENT

                                        1.  LSI-11 FAMILY PROCESSOR.
                                              11/03(LSI-11/02), 11/23(KDF11-A), 11/23B(KDF11-B)
                                        2.  DLV11 WITH I/O TYPE TERMINAL
                                        3.  DRV11B WITH LOOP BACK CABLE

                                2.2     STORAGE

                                        THE PROGRAM USES THE LOWER 4K OF MEMORY.

                                3.0     LOADING PROCEDURE
                                        -----------------

                                        IF USING PAPER TAPE READER FOLLOW THIS PROCEDURE:
                                        1.  ASSURE THAT THE LSI-11 IS IN THE ODT MICROCODE STATE.
                                        2.  LOAD THE LOW OR HIGH SPEED READER WITH THE ABSOLUTE LOADER
                                            TAPE.
                                        3.  TYPE THE READER'S CSR ADDRESS (177560-LOW OR 177550-HIGH)
                                            AND CHARACTER 'L'.
                                        4.  AFTER TAPE IS LOADED, LOAD THE DRV11B BINARY TAPE INTO THE
                                            READER AND TYPE THE CHARACTER 'P'.
                                        5.  IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STEPS 2 & 3),
                                            THEN ONLY THE STARTING ADDRESS OF THE ABSOLUTE LOADER AND
                                            THE CHARACTER 'G' NEED BE TYPED (WITH THE DRV11B BINARY TAPE
                                            IN THE APPROPRIATE READER).

                        4.0     STARTING PROCEDURE
                                ------------------

                                FOR PAPER TAPE MEDIA:
                                1.  MAKE SURE THE MAINTENANCE LOOP BACK CABLE IS INSERTED IN
                                    THE I/O CONNECTORS ON THE M7950 MODULE.
                                2.  MAKE SURE THE DEVICE BUS & VECTOR ADDRESSES AGREE WITH THE
                                    DEFAULT VALUES DEFINED IN SECTION 7.1.  IF NOT, CHANGE
                                    LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
                                3.  INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
                                4.  TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
                                5.  THE PROGRAM WILL RESPOND BY TYPING THE SOFTWARE SWITCH
                                    REGISTER CONTENTS AND ALLOWING THE USER TO CHANGE ITS
                                    CONTENTS BY ENTERING OCTAL SWITCH REGISTER DATA TERMINATED
                                    BY A CARRIAGE RETURN - SEE SECTION 5.0 FOR SWITCH REGISTER
```

```
                        OPTIONS.
                    6.  THE NEXT QUESTION ASKED IS ABOUT THE TYPE OF A PROCESSOR.
                        TYPE 'Y' WITH CARRIGE RETURN IF USING KDF11-B, 'N' OTHER-
                        WISE.

                    IF RUNNING UNDER XXDP+ MONITOR:
                    A)  DO 1., 2., 3. OF THE ABOVE.
                    B)  IN MONITOR MODE TYPE IN 'R NDRAA?'.
                    C)  DO 5. AND 6.

                    *****
                    IF USING LSI-11/23B(KDF11-B) AND WANT TO TEST DMA TRANSFERS TO
                    I/O PAGE PUT THE ADDRESS OF YOUR I/O INTERFACE CONTROL RE-
                    GISTER INTO LOCATION 1544 (IOPAGE). TO DO THIS PERMANENTLY USE
                    LOAD-MOD-DUMP PROCEDURE DEFINED IN 7.2. FOR TEMPORARY CHANGES
                    JUST MODIFY LOCATIONS AFTER LOADING THE PROGRAM.
                    NOTE: THIS TEST IS NOT GOING TO BE PERFOMED UNLESS THE ABOVE
                          MENTIONED LOCATIONS ARE MODIFIED.
```

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <5-0>
***SW07=1	000200	DEVICE IS KDF11-B

***DIAGNOSTIC WILL SET THIS SWITCH AUTOMATICALLY, IF THIS TYPE OF A PROCESSOR HAS BEEN CONFIRMED IN A DIALOGUE. IF RUNNING IN AUTOMATIC MODE (CHAINS UNDER XXDP OR APT) SEE SECTION 7.2

5.2 CONTROL

1. THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

6.2 ERROR DATA

*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	DRV11B BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED
ADRS	MEMORY ADDRESS OF DATA TRANSFER ON ERROR

```
                                *ALWAYS REPORTED
                        7.0     MISCELLANEOUS
                                -------------

                        7.1     DRV11B BUS & VECTOR ADDRESS MODIFICATION

                                MODIFY LOCATION '$BASE' IF BASE BUS ADDRESS IS NOT 174600.
                                MODIFY LOCATION '$VECT1' IF VECTOR ADDRESS IS NOT 210.

                                *NOTE:  USE THE LSI-11 ODT FACILITIES TO MODIFY THESE LOCATIONS
                                AFTER PROGRAM LOAD.  NO VECTOR ASSIGNMENT ABOVE 774 SHOULD BE
                                ALLOWED.

                        7.2     XXDP/APT NOTES

                                THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REF. 7.5)(REQUIRES 8K OR MORE).
                                THIS DIAGNOSTIC DOES SUPPORT "APT" AND HAS RUN UNDER IT.

                                IF THE PROCESSOR USED IS KDF11-B:
                                     FOR APT SET $SWREG BIT07 TO 1 (000200)
                                     FOR XXDP CHAINS SET LOCATION 176 (SWR) BIT07 TO 1
                                        TO DO THIS UNDER XXDP:
                                           1. R UPD2
                                           2. LOAD NDRAA?.BIC
                                           3. MOD 176
                                              THE TERMINAL WILL RESPOND: 176/000000
                                           4. NOW TYPE IN 200
                                           5. BE CAREFUL: AT THIS POINT IT IS NECESARY TO DELETE THE FILE
                                              FROM THE DISK.
                                              DEL DL0:NDRAA?.BIC (IF MEDIA IS RL ON DRIVE 0)
                                        6. DUMP NDRAA0.BIC
                                           TO BE SAFE, SKIP 5. AND DUMP THE FILE UNDER DIFFERENT NAME
                                           WITH .BIC EXTENSION.
                                     NOTE: THIS PROCEDURE ASSUMES THAT DIAGNOSTIC IS ON THE DISK
                                           FROM WHICH THE SYSTEM IS BOOTED. IF THIS IS UNTRUE
                                           IN 2. AND 6. THE OPERATOR HAVE TO SPECIFY THE DRIVE BEFORE
                                           THE NAME.

                        7.3     POWER FAIL

                                A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
                                WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
                                NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

                        7.4     MULTIPLE DRV11B INTERFACE TESTING

                                THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11B'S CONNECTED.
                                THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 DRV11B INTERFACES
                                WITH CONTIGUOUS BUS AND VECTOR ADDRESSES.  THIS IS ACCOMPLISHED
                                BY THE OPERATOR SETTING UP LOCATION '$DEVM' WITH A BIT MAP INDICATING WHAT
                                INTERFACES ARE TO TESTED.  I.E. BIT0=1 SAYS TEST 1ST DRV11B,
                                BIT1=1 SAYS TEST 2ND DRV11B, BIT2=1 SAYS TEST 3RD DRV11B, ETC..

                        7.5     RESTRICTIONS

                                IF THE SYSTEM ALSO INCLUDES AN "REV11" (DMA REFRESH),  THE DMA
```

```
                        REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.

                8.0     EXECUTION TIME
                        --------------

                        EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
                        TO ABOUT 90 SECONDS WITH ITERATIONS ENABLED WITH ONE DRV11B CONNECTED.
                        AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.
                9.0     PROGRAM TEST DESCRIPTIONS
                        -------------------------

                9.1     GENERAL

                        THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED
                        TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV11B DMA INTERFACE.
                        A HIGH DEGREE OF TESTING IS ACCOMPLISHED WITH THE AID OF THE
                        MAINTENANCE LOOP BACK CABLE PROVIDED FOR DIAGNOSTIC TESTING.
                        A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS
                        AT THE BEGINNING OF THE LISTING.  THE COMMENT FIELD WITHIN
                        EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.

                9.2     REGISTER TESTS

                        THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:

                        1.  WORD COUNT
                        2.  BUFFER ADDRESS
                        3.  COMMAND/STATUS
                        4.  DATA BUFFER

                9.3     BYTE ADDRESSING TESTS

                        1.  COMMAND/STATUS
                        2.  DATA BUFFER

                9.4     'FNCT' TO 'STAT' WRAP AROUND TEST

                9.5     READY INTERRUPT TEST

                9.6     NPR DATA TRANSFER TESTS

                        THE FOLLOWING NPR XFERS ARE CHECKED FOR CORRECT STATUS,
                        WORD COUNT, BUFFER ADDRESS & DATA:

                        1.  SINGLE 'DATI' XFER - FLOATING 1/0 PTRN
                        2.  SINGLE 'DATO' XFER - FLOATING 1/0 PTRN
                        3.  200 'DATI' XFERS - FLOATING 1/0 PTRN
                        4.  200 'DATO' XFERS - FLOATING 1/0 PTRN
                        5.  SINGLE 'DATI' XFER TO THE TTY PRINTER CSR
                            FOR KDF11-B PROCESSOR THE USER HAVE TO SELECT
                            CSR OF A PARTICULAR INTERFACE (DISK INTERFACES
                            ARE SUGGESTED). SOME OF THEM ARE:
                                RXV11   (RX01): 177170
                                RXV21   (RX02): 177170
                                RKV11-D (RK05): 177404
```

```
                                9.7     MAINT MODE NPR DATA TRANSFER TESTS

                                        1.  THAT MAINT MODE CONTROLS 'FNCT' BITS
                                        2.  200 MAINT MODE XFERS - CHECKING STATUS & DATA
                                        3.  200 MAINT MODE XFERS TO EACH 4K AVAILABLE MEM

                                9.8     BURST & NON-BURST MODE TESTS

                                        1.  THAT CPU IS LOCKED OUT IN BURST MODE
                                        2.  THAT CPU IS NOT LOCKED OUT IN NON-BURST MODE

                                9.9     'NEX' ERROR CONDITION TEST

                                10.0    REVISION HISTORY
                                        CVDRAC DIAGNOSTIC WAS MADE SPECIFIC TO 11/21 PROCESSOR AND
                                        IS CALLED CNDRAA0.
                                11.0    LISTING
                                        -------

                                %
```

```
                                .TITLE  MAINDEC-11-CNDRA-A DRV11B DMA INTERFACE DIAGNOSTIC
                                ;*COPYRIGHT (C) 1982
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY R. MOORE
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
000001                          $TN=1
160000                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
167400                          $SWR=167400
000300                          $SWRMK=300
000001                          $TN=1
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<5:0>
                                .SBTTL  BASIC DEFINITIONS

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

                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                                ;*****  THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
170000                          ODTST=  170000
                                ;*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
```

```
                                                        L 1
(1)             000007          PC=     %7              ;;PROGRAM COUNTER
(1)
(1)                             ;*PRIORITY LEVEL DEFINITIONS
(1)             000000          PR0=    0               ;;PRIORITY LEVEL 0
(1)             000040          PR1=    40              ;;PRIORITY LEVEL 1
(1)             000100          PR2=    100             ;;PRIORITY LEVEL 2
(1)             000140          PR3=    140             ;;PRIORITY LEVEL 3
(1)             000200          PR4=    200             ;;PRIORITY LEVEL 4
(1)             000240          PR5=    240             ;;PRIORITY LEVEL 5
(1)             000300          PR6=    300             ;;PRIORITY LEVEL 6
(1)             000340          PR7=    340             ;;PRIORITY LEVEL 7
(1)
(1)                             ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)             100000          SW15=   100000
(1)             040000          SW14=   40000
(1)             020000          SW13=   20000
(1)             010000          SW12=   10000
(1)             004000          SW11=   4000
(1)             002000          SW10=   2000
(1)             001000          SW09=   1000
(1)             000400          SW08=   400
(1)             000200          SW07=   200
(1)             000100          SW06=   100
(1)             000040          SW05=   40
(1)             000020          SW04=   20
(1)             000010          SW03=   10
(1)             000004          SW02=   4
(1)             000002          SW01=   2
(1)             000001          SW00=   1
(1)                             .EQUIV  SW09,SW9
(1)                             .EQUIV  SW08,SW8
(1)                             .EQUIV  SW07,SW7
(1)                             .EQUIV  SW06,SW6
(1)                             .EQUIV  SW05,SW5
(1)                             .EQUIV  SW04,SW4
(1)                             .EQUIV  SW03,SW3
(1)                             .EQUIV  SW02,SW2
(1)                             .EQUIV  SW01,SW1
(1)                             .EQUIV  SW00,SW0
(1)
(1)                             ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)             100000          BIT15=  100000
(1)             040000          BIT14=  40000
(1)             020000          BIT13=  20000
(1)             010000          BIT12=  10000
(1)             004000          BIT11=  4000
(1)             002000          BIT10=  2000
(1)             001000          BIT09=  1000
(1)             000400          BIT08=  400
(1)             000200          BIT07=  200
(1)             000100          BIT06=  100
(1)             000040          BIT05=  40
(1)             000020          BIT04=  20
(1)             000010          BIT03=  10
(1)             000004          BIT02=  4
(1)             000002          BIT01=  2
```

```
                        000001                  BIT00=  1
(1)                                             .EQUIV  BIT09,BIT9
(1)                                             .EQUIV  BIT08,BIT8
(1)                                             .EQUIV  BIT07,BIT7
(1)                                             .EQUIV  BIT06,BIT6
(1)                                             .EQUIV  BIT05,BIT5
(1)                                             .EQUIV  BIT04,BIT4
(1)                                             .EQUIV  BIT03,BIT3
(1)                                             .EQUIV  BIT02,BIT2
(1)                                             .EQUIV  BIT01,BIT1
(1)                                             .EQUIV  BIT00,BIT0
(1)
(1)                                             ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)                     000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)                     000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)                     000014                  TBITVEC=14              ;;"T" BIT
(1)                     000014                  TRTVEC= 14              ;;TRACE TRAP
(1)                     000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)                     000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)                     000024                  PWRVEC= 24              ;;POWER FAIL
(1)                     000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)                     000034                  TRAPVEC=34              ;;"TRAP" TRAP
(1)                     000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)                     000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
(1)                                             ;*****  THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(1)                     000100                  LKVEC=  100             ;;LINE CLOCK VECTOR
(1)                     000140                  BRKVEC= 140             ;;BREAK VECTOR
(1)                     000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
6060                    174600                  ABASE=  174600          ;BASE DRV11B BUS ADRS EQUATE
6061                    000210                  AVECT1= 000210          ;BASE DRV11B VECTOR ADRS EQUATE -
6062                    000001                  ADEVM=  1               ;DEFAULT TO ONE DRV11B
6063                    106427                  MTPS=106427             ;INSTR EQUATE THAT MOVES BYTE TO PSW
6064    000000          177522                  TMAIN: 177522           ;MAINTENCE REGISTER(FOR USE WITH KDF11-B)
6065                                            .SBTTL  TRAP CATCHER
(1)
(1)                     000000                          .=0
(1)                                             ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)                                             ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)                                             ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)                     000174                          .=174
(1)     000174  000000                          DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)     000176  000000                          SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
(1)                                             .SBTTL  STARTING ADDRESS(ES)
(1)     000200  000137  001550                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
6066                    000100                  .=100
6067    000100  000104  000200  000002         .WORD   104,200,2       ;IF 'B EVENT' ON Q BUS IS CONNECTED
6068                                                                    ;IGNORE IT'S INTERRUPT - JUST DO A RTI
```

```
6070                                    .SBTTL  ACT11 HOOKS
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;HOOKS REQUIRED BY ACT11
 (1)          000106                            $SVPC=.                 ;SAVE PC
 (1)          000046                            .=46
 (1)  000046  010056                            $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
 (1)          000052                            .=52
 (1)  000052  000000                            .WORD   0               ;;2)SET LOC.52 TO ZERO
 (1)          000106                            .=$SVPC                 ;; RESTORE PC
6071          001000                            .=1000
6072                                    .SBTTL  APT PARAMETER BLOCK
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
 (2)                                    ;;*****************************************************************
 (1)          001000                            .$X=.   ;;SAVE CURRENT LOCATION
 (1)          000024                            .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
 (1)  000024  000200                            200     ;;FOR APT START UP
 (1)          000044                            .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
 (1)  000044  001000                            $APTHDR ;;POINT TO APT HEADER BLOCK
 (1)          001000                            .=.$X   ;;RESET LOCATION COUNTER
 (2)                                    ;;*****************************************************************
 (1)                                    ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 (1)                                    ;INTERFACE SPEC.
 (1)
 (1)  001000                            $APTHD:
 (1)  001000  000000                    $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 (1)  001002  001174                    $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
 (1)  001004  000031                    $TSTM:  .WORD   25.     ;;RUN TIM OF LONGEST TEST
 (1)  001006  000006                    $PASTM: .WORD   6.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 (1)  001010  000144                    $UNITM: .WORD   100.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
 (1)  001012  000052                            .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2
                                ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001162  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001164  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001170  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001171  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001172  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*******************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*******************************************************************
                                .EVEN
001174                  $MAIL:                          ;;APT MAILBOX
001174  000000          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001176  000000          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001200  000000          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001202  000000          $PASS:  .WORD   APASS           ;;PASS COUNT
001204  000000          $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001206  000000          $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
```

```
(2)  001210  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001212  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001214                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001214     000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001215     000                $ENVM:  .BYTE   AENVM
(2)                                ;;ENVIRONMENT MODE BITS
(2)  001216  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001220  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001222  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                             11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                             11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001224     000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001225     000                $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                             900 NSEC CORE=001
(2)                                ;*                             300 NSEC BIPOLAR=002
(2)                                ;*                             500 NSEC MOS=003
(2)  001226  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)  001230     000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001231     000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001232  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001234     000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001235     000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001236  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001240     000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001241     000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001242  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001244  000210                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001246  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001250  174600                $BASE:  .WORD   ABASE
(2)                                ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001252  000001                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001254  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001256  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(2)  001260  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
(2)  001262  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
(2)  001264  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
(2)  001266  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
(2)  001270  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
(2)  001272  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
(2)  001274  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
(2)  001276  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
(2)  001300  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
(2)  001302  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
(2)  001304  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
(2)  001306  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
(2)  001310  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
(2)  001312  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
(2)  001314  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
(2)  001316  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
(2)
```

```
    (2)
    (2)  001320                         $ETEND:
    (2)
```

```
                                .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


001320                          $ERRTB:
                                ;ERROR  1
014066                                  EM1             ;REG TIMEOUT ER
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0

                                ;ERROR  2
014105                                  EM2             ;REG READ/WRITE ER
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0

                                ;ERROR  3
014127                                  EM3             ;BUS RESET ER
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0

                                ;ERROR  4
014144                                  EM4             ;FNCT BITS FAILED TO SET STAT BITS
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0

                                ;ERROR  5
014206                                  EM5             ;READY INTR FAILURE
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0

                                ;ERROR  6
014231                                  EM6             ;READY CLR OR SET ER
014757                                  DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
015130                                  DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
000000                                  0
```

```
                                ;ERROR  7
001400  014255                          EM7             ;STATUS ER ON XFER
001402  014757                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001404  015130                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001406  000000                          0

                                ;ERROR  10
001410  014277                          EM10            ;WORD COUNT ER ON XFER
001412  014757                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001414  015130                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001416  000000                          0

                                ;ERROR  11
001420  014325                          EM11            ;BUFFER ADRS ER ON XFER
001422  014757                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001424  015130                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001426  000000                          0

                                ;ERROR  12
001430  014354                          EM12            ;DATA ER FROM MEM
001432  015024                          DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001434  015144                          DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001436  000000                          0

                                ;ERROR  13
001440  014375                          EM13            ;DATA ER TO MEM
001442  015024                          DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001444  015144                          DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001446  000000                          0

                                ;ERROR  14
001450  014414                          EM14            ;SINGLE CYCLE OFF DID NOT LOCK OUT CPU
001452  015101                          DH3             ;ERRPC  TSTNUM  BUSADR
001454  015162                          DT3             ;$ERRPC TSTNUM  $BDADR
001456  000000                          0

                                ;ERROR  15
001460  014462                          EM15            ;SINGLE CYCLE ON LOCKED OUT CPU
001462  015101                          DH3             ;ERRPC  TSTNUM  BUSADR
001464  015162                          DT3             ;$ERRPC TSTNUM  $BDADR
001466  000000                          0

                                ;ERROR  16
001470  014650                          EM16            ;NEX LOGIC ER
001472  014757                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001474  015130                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001476  000000                          0

                                ;ERROR  17
001500  014665                          EM17            ;CYCLE FAILED TO CLK DBR (IN)
001502  014757                          DH1             ;ERRPC  TSTNUM  BDADR   GDDAT   BDDAT
001504  015130                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001506  000000                          0
```

```
                                        ;ERROR  20
001510  014722                                  EM20            ;DATA ER FROM I/O PAGE (XCSR)
001512  015024                                  DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001514  015144                                  DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001516  000000                                  0

                                        ;DRV11B BUS REGISTER ADDRESS POINTERS

001520  174600                          DRVWCR: ABASE           ;WORD COUNT
001522  174602                          DRVBAR: ABASE+2         ;BUFFER ADDRESS
001524  174604                          DRVCSR: ABASE+4         ;COMMAND/STATUS
001526  174606                          DRVDBR: ABASE+6         ;DATA BUFFER

                                        ;DRV11B VECTOR ADDRESS POINTERS

001530  000210                          DRVCT0:AVECT1           ;READY, NEX & INCOMPLETE DATIO VECTOR
001532  000212                          DRVCT2: AVECT1+2                ;NEW PSW ON INTR

                                        ;COMMON PROGRAM LOCATION(S)

001534  000000                          TSTNUM: 0               ;CONTAINS TEST NUMBER ON ERROR
001536  000001                          DMAP:   1               ;DEVICE MAP - EA BIT SAYS TEST THAT DRV11B
001540  000000                          CORSZ:  0               ;CONTAINS 1ST NON-EXISTANT MEM ADRS
001542  015316                          DBUFP:  DBUF            ;CONTAINS CURRENT 4K NPR BUFFER ADRS
001544  000000                          IOPAGE: 0               ;CONTAINS ADDRESS FOR I/O TRANSFERS
001546  000000                          KDF:    0               ;IF KDF11-B
```

```
                                                        .SBTTL  PROGRAM START
001550                                          START:
                                                .SBTTL  INITIALIZE THE COMMON TAGS
                                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001550  012706  001100                                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001554  005026                                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001556  022706  001140                                  CMP     #SWR,R6 ;;DONE?
001562  001374                                          BNE     .-6             ;;LOOP BACK IF NO
001564  012706  001100                                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                                ;;INITIALIZE A FEW VECTORS
001570  012737  012374  000020                          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001576  012737  000300  000022                          MOV     #PR6,@#IOTVEC+2 ;;LEVEL 6
001604  012737  012032  000030                          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001612  012737  000300  000032                          MOV     #PR6,@#EMTVEC+2 ;;LEVEL 6
                                                ;;BIT02
001620  012737  014004  000034                          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001626  012737  000300  000036                          MOV     #PR6,@#TRAPVEC+2;LEVEL 6
001634  012737  013600  000024                          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001642  012737  000300  000026                          MOV     #PR6,@#PWRVEC+2 ;;LEVEL 6
001650  005037  001160                                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001654  005037  001162                                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001660  112737  000001  001115                          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001666  012737  001666  001106                          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001674  012737  001674  001110                          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001702  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001706  012737  001742  000004                  MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001714  012737  177570  001140                  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001722  012737  177570  001142                  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001730  022777  177777  177202                  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001736  001012                                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
001740  000403                                  BR      65$             ;;BRANCH IF NO TIMEOUT
001742  012716  001750                  64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001746  000002                                  RTI
001750  012737  000176  001140          65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001756  012737  000174  001142                  MOV     #DISPREG,DISPLAY
001764  012637  000004                  66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001770  005037  001202                          CLR     $PASS           ;;CLEAR PASS COUNT
001774  132737  000200  001215                  BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
002002  001403                                  BEQ     67$             ;;YES,USE NON-APT SWITCH
002004  012737  001216  001140                  MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
002012                                  67$:
002012  012700  001520                  START1: MOV     #DRVWCR,R0      ;SET UP REG ADRS POINTERS
002016  013701  001250                          MOV     $BASE,R1        ;GET BASE ADRS
002022  010120                          SETUP2: MOV     R1,(R0)+        ;LOAD EM
002024  062701  000002                          ADD     #2,R1           ;
002030  022700  001530                          CMP     #DRVDBR+2,R0    ;ALL DONE?
002034  001372                                  BNE     SETUP2          ;BR IF NOT
002036  012700  001530                          MOV     #DRVCTO,R0      ;SET UP DRV11B VECTOR ADRS POINTER
002042  013701  001244                          MOV     $VECT1,R1       ;GET BASE VECTOR ADRS
002046  042701  170000                          BIC     #170000,R1      ;CLR OUT PRIORITY BITS
002052  010120                          SETUP3: MOV     R1,(R0)+        ;
002054  062701  000002                          ADD     #2,R1           ;POINT TO NEXT
```

```
  6209  002060  022700  001534                    CMP     #DRVCT2+2,R0    ;ALL DONE?
  6210  002064  001372                            BNE     SETUP3          ;BR IF NOT
  6211                                    .SBTTL  TYPE PROGRAM NAME
   (1)                                    ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
   (1)  002066  005227  177777                    INC     #-1             ;;FIRST TIME?
   (1)  002072  001052                            BNE     64$             ;;BRANCH IF NO
   (1)  002074  104401  002142                    TYPE    ,65$            ;;TYPE ASCIZ STRING
   (2)                                    .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
   (2)  002100  005737  000042                    TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
   (2)  002104  001012                            BNE     66$             ;;BRANCH IF YES
   (2)  002106  123727  001214  000001            CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
   (2)  002114  001406                            BEQ     66$             ;;BRANCH IF YES
   (2)  002116  023727  001140  000176            CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
   (2)  002124  001005                            BNE     67$             ;;BRANCH IF NO
   (2)  002126  104406                            GTSWR                   ;;GET SOFT-SWR SETTINGS
   (2)  002130  000403                            BR      67$
   (2)  002132  112737  000001  001134  66$:      MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
   (2)  002140                            67$:
   (1)  002140  000427                            BR      64$             ;;GET OVER THE ASCIZ
   (1)                                    ;;65$:  .ASCIZ  <CRLF>#MD-11-CNDRA-A  DRV11B DMA INTERFACE DIAG  #<CRLF>
   (1)  002220                            64$:
  6212  002220  005227  177777                    INC     #-1             ;FIRST PASS?
  6213  002224  001022                            BNE     100$            ;BRANCH IF NO
  6214  002226  122737  000001  001134            CMPB    #1,$AUTOB       ;MANUAL INTERVENTION PERMITTED?
  6215  002234  001416                            BEQ     100$            ;BR IF NO
  6216  002236  104401  015172                    TYPE,   KDF11B          ;ASK ABOUT KDF11-B
  6217  002242  104410                            RDCHR                   ;READ ANSWER
  6218  002244  012637  001546                    MOV     (SP)+,KDF       ;STORE ANSWER
  6219  002250  104401  001546                    TYPE,   KDF             ;ECHO ANSWER
  6220  002254  123727  001546  000131            CMPB    KDF,#131        ;IS IT KDF11-B?
  6221  002262  001003                            BNE     100$            ;BRANCH IF NOT
  6222  002264  152777  000200  176646            BISB    #BIT07,@SWR     ;IF YES,SET SWREG
  6223  002272  005737  001540          100$:     TST     CORSZ           ;TEST IF FIRST PASS
  6224  002276  001002                            BNE     CORSZR          ;BR IF NOT
  6225  002300  104401  014521                    TYPE,   WARN            ;TELL THE OPERATOR TO TURN OFF DMA REFRESH
  6226                                    ;;**************************************************************************
  6227                                    ;LET'S SEE HOW MUCH MEM WE HAVE
  6228                                    ;;**************************************************************************
  6229  002304  012700  020000          CORSZR: MOV     #20000,R0       ;USE R0 TO LOOK
  6230  002310  012737  002322  000004            MOV     #2$,@#ERRVEC    ;SET UP TIME OUT RETURN ADRS
  6231  002316  005720                  1$:       TST     (R0)+           ;TAKE A LOOK
  6232  002320  000776                            BR      1$              ;UNTIL TIMEOUT
  6233  002322  042700  017777          2$:       BIC     #17777,R0       ;POINT TO 1ST NON-EXSISTANT 4K BLK
  6234  002326  010037  001540                    MOV     R0,CORSZ        ;SAVE FOR LATER
  6235  002332  012737  000006  000004            MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE VECTOR
  6236  002340  012737  015316  001542            MOV     #DBUF,DBUFP     ;INITIALIZE TO LOWEST 4K
  6237  002346  012737  000000  001206            MOV     #0,$UNIT        ;SET UP UNIT COUNT
  6238  002354  013737  001252  001536            MOV     $DEVM,DMAP      ;GET THE # & POSITION OF DRV11B'S
  6239  002362  042737  177400  001536            BIC     #177400,DMAP    ;UP TO 8 ONLY
  6240  002370  001406                            BEQ     RESTRT          ;GO CONTINUE AS IF SOMETHING WAS SELECTED
  6241  002372  032737  000001  001536            BIT     #1,DMAP         ;IS 1ST DRV11B SELECTED?
  6242  002400  001002                            BNE     RESTRT          ;BR IF SO
  6243  002402  000137  007674                    JMP     NXDEV1          ;NO - GO ADVANCE BASE DRV11B ADDRESSES
  6244  002406  106427  000200          RESTRT: MTPS    #200            ;SET PRIORITY TO HIGHEST LEVEL
  6245  002412  012706  001100                    MOV     #STACK,SP       ;ALWAYS RESET STACK PTR
  6246  002416  013737  001206  001204            MOV     $UNIT,$DEVCT    ;LOAD APT COUNTER
```

```
 6247  002424  013700  001206                MOV     $UNIT,R0        ;MAKE AN INDEX
 6248  002430  006300                        ASL     R0              ; VALUE
 6249  002432  013760  001520  001260        MOV     DRVWCR,$DDW0(R0)        ;SAVE THE BUS ADDRESS
 6250  002440  000005                        RESET                   ;INITIALIZE DRV11B BEFORE TESTING
 6251                                  ;;*****************************************************************
  (3)                                  ;*TEST 1        TEST THAT ALL DRV11B REGS ARE ACCESSIBLE
  (3)                                  ;;*****************************************************************
  (2)  002442  000240                  TST1:   <NOP>
  (2)
  (1)  002444  012737  002460  001106          MOV     #10$,$LPADR     ;;SET SCOPE LOOP ADDRESS
  (2)  002452  012737  000001  001200          MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 6252  002460  112737  000001  001102  10$:    MOVB    #1,$TSTNM       ;SET TO TEST #1
 6253  002466  012737  002522  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6254  002474  005037  001124                CLR     $GDDAT          ;NO DATA COMPARE
 6255  002500  005037  001126                CLR     $BDDAT          ;NO DATA COMPARE
 6256  002504  012737  002540  000004        MOV     #2$,@#ERRVEC    ;SET UP TIMEOUT RETURN ADDRS
 6257  002512  013700  001520                MOV     DRVWCR,R0       ;SET UP 1ST DRV11 BUS ADRS
 6258  002516  012701  000004                MOV     #4,R1           ;SET UP REG COUNT
 6259  002522  010037  001122  1$:     MOV     R0,$BDADR       ;SET UP CURRENT DRV BUS ADRS
 6260  002526  005710                        TST     (R0)            ;SEE IF THERE
 6261  002530  005720                        TST     (R0)+           ;BUMP TO NEXT
 6262  002532  005301                        DEC     R1              ;COUNT 4 OF THEM
 6263  002534  001403                        BEQ     3$              ;BR IF ALL DONE
 6264  002536  000771                        BR      1$              ;TRY NEXT
 6265  002540  022626                2$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
 6266  002542  104001                        ERROR   1               ;BUS ADRS INDICATED DID NOT RESPOND
 6267  002544  012737  000006  000004  3$:     MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE LOC 4
 6268
 6269                                  ;;*****************************************************************
  (3)                                  ;*TEST 2        TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
  (3)                                  ;;*****************************************************************
  (2)  002552  000004                  TST2:   SCOPE
  (2)
 6270  002554  012737  002600  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6271  002562  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WC REG ADRS
 6272  002570  005000                        CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
 6273  002572  012737  177776  001124        MOV     #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
 6274  002600  013777  001124  176712  1$:     MOV     $GDDAT,@DRVWCR  ;LD WC
 6275  002606  017737  176706  001126        MOV     @DRVWCR,$BDDAT  ;READ IT BACK
 6276  002614  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
 6277  002622  001401                        BEQ     2$              ;BR IF SO
 6278  002624  104002                        ERROR   2               ;WORD COUNT WRITE/READ FAILURE
 6279  002626  005137  001124          2$:     COM     $GDDAT          ;COMPELEMENT ZERO
 6280  002632  005100                        COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
 6281  002634  001361                        BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
 6282  002636  006337  001124                ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
 6283  002642  005237  001124                INC     $GDDAT          ;KEEP LSB SET
 6284  002646  103754                        BCS     1$              ;AGAIN TILL ALL PATRNS DONE
 6285
 6286                                  ;;*****************************************************************
  (3)                                  ;*TEST 3        TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
  (3)                                  ;;*****************************************************************
  (2)  002650  000004                  TST3:   SCOPE
  (2)
 6287  002652  012737  002676  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6288  002660  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
```

```
 6289  002666  005000                          CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
 6290  002670  012737  177774  001124          MOV     #-4,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
 6291  002676  013777  001124  176616  1$:     MOV     $GDDAT,@DRVBAR  ;LD BA
 6292  002704  017737  176612  001126          MOV     @DRVBAR,$BDDAT  ;READ IT BACK
 6293  002712  042737  000001  001126          BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
 6294  002720  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6295  002726  001401                          BEQ     2$              ;BR IF SO
 6296  002730  104002                          ERROR   2               ;BUS ADRS WRITE/READ FAILURE
 6297  002732  005137  001124          2$:     COM     $GDDAT          ;COMPLEMENT ZERO
 6298  002736  042737  000001  001124          BIC     #BIT00,$GDDAT   ;BIT 00 NOT INVOLVED
 6299  002744  005100                          COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
 6300  002746  001353                          BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
 6301  002750  006337  001124                  ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
 6302  002754  103004                          BCC     TST4            ;;NEXT TEST IF BIT 15 DONE
 6303  002756  062737  000002  001124          ADD     #2,$GDDAT       ;KEEP ADDR LSB SET
 6304  002764  000744                          BR      1$              ;AGAIN TILL ALL PATTERNS DONE
 6305
 6306                                  ;;*****************************************************************
   (3)                                 ;*TEST 4        TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
   (3)                                 ;;*****************************************************************
   (2)  002766  000004                  TST4:   SCOPE
   (2)
 6307  002770  012737  003014  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6308  002776  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DB REG ADRS
 6309  003004  005000                          CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
 6310  003006  012737  177776  001124          MOV     #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
 6311  003014  013777  001124  176504  1$:     MOV     $GDDAT,@DRVDBR  ;LD DB
 6312  003022  017737  176500  001126          MOV     @DRVDBR,$BDDAT  ;READ IT BACK
 6313  003030  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6314  003036  001401                          BEQ     2$              ;BR IF SO
 6315  003040  104002                          ERROR   2               ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
 6316  003042  005137  001124          2$:     COM     $GDDAT          ;COMPLEMENT ZERO
 6317  003046  005100                          COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
 6318  003050  001361                          BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
 6319  003052  006337  001124                  ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
 6320  003056  005237  001124                  INC     $GDDAT          ;KEEP LSB SET
 6321  003062  103754                          BCS     1$              ;AGAIN TILL ALL PATRNS DONE
 6322
 6323                                  ;;*****************************************************************
   (3)                                 ;*TEST 5        TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE
   (3)                                 ;;*****************************************************************
   (2)  003064  000004                  TST5:   SCOPE
   (2)
 6324  003066  013700  001526                  MOV     DRVDBR,R0       ;GET DB REG ADRS
 6325  003072  010037  001122                  MOV     R0,$BDADR       ;SET UP DB REG ADRS
 6326  003076  005010                          CLR     (R0)            ;ZERO DATA BUFFER REG
 6327  003100  012737  177177  001124          MOV     #177177,$GDDAT  ;LD EXPECTED
 6328  003106  012737  077776  001126          MOV     #77776,$BDDAT   ;SEND DATA FROM "BDDAT"
 6329  003114  153760  001126  000001          BISB    $BDDAT,1(R0)    ;LOAD HI BYTE DB
 6330  003122  153710  001127                  BISB    $BDDAT+1,(R0)   ;LOAD LO BYTE DB
 6331  003126  011037  001126                  MOV     (R0),$BDDAT     ;READ IT BACK
 6332  003132  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6333  003140  001401                          BEQ     TST6            ;;NEXT TEST IF SO
 6334  003142  104002                          ERROR   2               ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG
 6335
 6336                                  ;;*****************************************************************
```

```
   (3)                                  ;*TEST 6        TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
   (3)                                  ;;*************************************************************************
   (2)  003144  000004                  TST6:   SCOPE
   (2)
   (1)  003146  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  6337  003154  005037  001124          CLR     $GDDAT          ;LD EXPECTED
  6338  003160  012777  177777  176332  MOV     #-1,@DRVWCR     ;SET ALL BITS - WC REG
  6339  003166  012777  177776  176326  MOV     #-2,@DRVBAR     ;SET ALL BITS - BUS ADRS REG
  6340  003174  012777  177777  176324  MOV     #-1,@DRVDBR     ;SET ALL BITS - DB OUT REG
  6341  003202  000005                  RESET                   ;DO A BUS RESET
  6342  003204  017737  176310  001126  MOV     @DRVWCR,$BDDAT  ;READ WC REG
  6343  003212  001404                  BEQ     1$              ;BR IF CLRED
  6344  003214  013737  001520  001122  MOV     DRVWCR,$BDADR   ;SET UP WC REG ADRS
  6345  003222  104003                  ERROR   3               ;RESET FAILED TO CLR WC REG
  6346  003224  017737  176272  001126  1$:     MOV     @DRVBAR,$BDDAT  ;READ BUS ADRS REG
  6347  003232  042737  000001  001126  BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
  6348  003240  001404                  BEQ     2$              ;BR IF CLRED
  6349  003242  013737  001522  001122  MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
  6350  003250  104003                  ERROR   3               ;RESET FAILED TO CLR BUS ADRS REG
  6351  003252  017737  176250  001126  2$:     MOV     @DRVDBR,$BDDAT  ;READ DATA BUFFER REG
  6352  003260  001404                  BEQ     TST7            ;;NEXT TEST IF CLRED
  6353  003262  013737  001526  001122  MOV     DRVDBR,$BDADR   ;SET UP DB REG ADRS
  6354  003270  104003                  ERROR   3               ;RESET FAILED TO CLR DATA BUFFER OUT REG
  6355
  6356                                  ;;*************************************************************************
   (3)                                  ;*TEST 7        TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
   (3)                                  ;;*************************************************************************
   (2)  003272  000004                  TST7:   SCOPE
   (2)
   (1)  003274  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  6357  003302  106427  000200          MTPS    #200            ;DON'T WANT ANY INTRS
  6358  003306  004537  010112          JSR     R5,SETVEC       ;SET UP INTR RETURN ADRS IN CASE
  6359  003312  003410                  3$                      ;RETURN TO 3$ ON ILLEGAL INTR
  6360  003314  013737  003334  001110  MOV     1$,$LPERR       ;SET UP SCOPE LOOP ADRS
  6361  003322  013737  001524  001122  MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
  6362  003330  012700  160000          MOV     #160000,R0      ;START AT 0 - HI BITS FOR NOISE
  6363  003334  010037  001124  1$:     MOV     R0,$GDDAT       ;LD EXPECTED
  6364  003340  042737  167201  001124  BIC     #167201,$GDDAT  ;MASK TO WRITEABLE BITS
  6365  003346  010077  176152          MOV     R0,@DRVCSR      ;LD CSR
  6366  003352  017737  176146  001126  MOV     @DRVCSR,$BDDAT  ;READ IT BACK
  6367  003360  042737  007200  001126  BIC     #7200,$BDDAT    ;DON'T LOOK AT STAT & RDY BITS
  6368  003366  023737  001124  001126  CMP     $GDDAT,$BDDAT   ;CORRECT?
  6369  003374  001401                  BEQ     2$              ;BR IF SO
  6370  003376  104002                  ERROR   2               ;CONTROL/STATUS REG WRITE/READ FAILURE
  6371  003400  062700  000002          2$:     ADD     #2,R0           ;ADVANCE COUNT PATTERN
  6372  003404  001353                  BNE     1$              ;WRITE NEXT PATTERN IF NOT ALL TESTED
  6373  003406  000413                  BR      4$              ;GO RESTORE VECTOR
  6374  003410  022626                  3$:     CMP     (SP)+,(SP)+    ;FIX STACK - SHOULD NOT HAVE INTR'ED
  6375  003412  052737  000200  001124  BIS     #200,$GDDAT     ;CORRECT EXPECTED
  6376  003420  017737  176100  001126  MOV     @DRVCSR,$BDDAT  ;READ CSR
  6377  003426  042737  007000  001126  BIC     #7000,$BDDAT    ;DON'T WANT 'STAT' BITS
  6378  003434  104005                  ERROR   5               ;CPU FAILED TO LOCK OUT DRV11B INTR REQ
  6379  003436  004737  010132          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
  6380
  6381                                  ;;*************************************************************************
   (3)                                  ;*TEST 10       TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
```

```
                                ;;*****************************************************************
                        TST10:  SCOPE
                                ;;*****************************************************************
                                ;;
                                ;;DO 10 ITERATIONS
003444  012737  000010  001160          MOV     #10,$TIMES
                                ;;SET UP CSR ADRS
003452  013737  001524  001122          MOV     DRVCSR,$BDADR
                                ;;LD EXPECTED
003460  012737  000200  001124          MOV     #200,$GDDAT
                                ;;LD ALL CSR BITS
003466  012777  177776  176030          MOV     #-2,@DRVCSR
                                ;;DO A BUS RESET
003474  000005                          RESET
                                ;;READ CSR
003476  017737  176022  001126          MOV     @DRVCSR,$BDDAT
                                ;;CORRECT?
003504  023737  001124  001126          CMP     $GDDAT,$BDDAT
                                ;;NEXT TEST IF CSR CLRED & READY SET
003512  001401                          BEQ     TST11
                                ;;RESET FAILED TO SET UP THE CSR
003514  104003                          ERROR   3
                                ;;*****************************************************************
                                ;*TEST 11        TEST THAT THE CSR IS BYTE ADDRESSABLE
                                ;;*****************************************************************
                        TST11:  SCOPE
                                ;;GET CSR ADRS
003520  013700  001524                  MOV     DRVCSR,R0
                                ;;SET UP CSR ADRS
003524  010037  001122                  MOV     R0,$BDADR
                                ;;ZERO CSR
003530  005010                          CLR     (R0)
                                ;;LD EXPECTED
003532  012737  010300  001124          MOV     #10300,$GDDAT
                                ;;SEND DATA FROM "BDDAT" - USE MAIN + IE
003540  012737  040020  001126          MOV     #40020,$BDDAT
                                ;;LOAD HI BYTE CSR
003546  153760  001126  000001          BISB    $BDDAT,1(R0)
                                ;;LOAD LO BYTE CSR
003554  153710  001127                  BISB    $BDDAT+1,(R0)
                                ;;READ IT BACK
003560  011037  001126                  MOV     (R0),$BDDAT
                                ;;CORRECT?
003564  023737  001124  001126          CMP     $GDDAT,$BDDAT
                                ;;BR IF SO
003572  001401                          BEQ     1$
                                ;;DATA ERROR ON BYTE ADDRESSING THE CSR
003574  104002                          ERROR   2
                                ;;ZERO CSR BEFORE ADVANCING
003576  005010                  1$:     CLR     (R0)
                                ;;*****************************************************************
                                ;*TEST 12        TEST THAT THE 3 "FNCT" BITS CONTROL THE 3 "STAT" BITS (COUNT PTRN)
                                ;;*****************************************************************
                        TST12:  SCOPE
                                ;;SET UP SCOPE LOOP ADRS
003602  012737  003630  001110          MOV     #1$,$LPERR
                                ;;SET UP CSR ADRS
003610  013737  001524  001122          MOV     DRVCSR,$BDADR
                                ;;LD EXPECTED
003616  012737  007000  001124          MOV     #7000,$GDDAT
                                ;;R0 CONTAINS "FNCT" BITS WRITTEN
003624  012700  000016                  MOV     #16,R0
                                ;;WRITE INTO "FNCT" BITS
003630  010077  175670          1$:     MOV     R0,@DRVCSR
                                ;;READ BACK THRU "STAT" BITS
003634  017737  175664  001126          MOV     @DRVCSR,$BDDAT
                                ;;MASK TO "STAT" BITS ONLY
003642  042737  170777  001126          BIC     #170777,$BDDAT
                                ;;CORRECT?
003650  023737  001124  001126          CMP     $GDDAT,$BDDAT
                                ;;BR IF SO
003656  001401                          BEQ     2$
                                ;;"FNCT" BITS FAILED TO SET "STAT" BITS (LOOP BACK)
003660  104004                          ERROR   4
                                ;;CHANGE TO NEXT EXPECTED
003662  162737  001000  001124  2$:     SUB     #1000,$GDDAT
                                ;;DECREASE COUNT PATTERN
003670  162700  000002                  SUB     #2,R0
                                ;;DO AGAIN UNTIL 0 TESTED
003674  100355                          BPL     1$
                                ;;*****************************************************************
                                ;*TEST 13        TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0
                                ;;*****************************************************************
                        TST13:  SCOPE
                                ;;DO 10 ITERATIONS
003700  012737  000010  001160          MOV     #10,$TIMES
```

```
 6421  003706  106427  000200            MTPS    #200            ;DONT WANT INTR YET
 6422  003712  000005                    RESET                   ;SET THE READY FLAG BY INIT
 6423  003714  012777  003774  175606    MOV     #1$,@DRVCTO     ;SET UP PREMATURE INTR RETURN ADRS
 6424  003722  013737  001524  001122    MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6425  003730  012737  000300  001124    MOV     #300,$GDDAT     ;LD EXPECTED (READY + IE)
 6426  003736  106427  000000            MTPS    #0              ;ALLOW AN INTR
 6427  003742  021616                    CMP     (SP),(SP)       ;STALL
 6428  003744  012777  004010  175556    MOV     #2$,@DRVCTO     ;SET UP EXPECTED INTR RETURN ADRS
 6429  003752  052777  000100  175544    BIS     #BIT6,@DRVCSR   ;ENABLE THE EXPECTED INTERRUPT
 6430  003760  021616                    CMP     (SP),(SP)       ;STALL
 6431  003762  017737  175536  001126    MOV     @DRVCSR,$BDDAT  ;GET THE CSR
 6432  003770  104005                    ERROR   5               ;READY FAILED TO CAUSE AN INTERRUPT
 6433  003772  000417                    BR      3$              ;GO RESTORE VECTOR
 6434  003774  022626            1$:     CMP     (SP)+,(SP)+     ;SHOULD NEVER GET HERE - IE NOT WORKING?
 6435  003776  017737  175522  001126    MOV     @DRVCSR,$BDDAT  ;GET THE CSR
 6436  004004  104005                    ERROR   5               ;READY INTERRUPTED WITHOUT THE IE BIT
 6437  004006  000411                    BR      3$              ;GO RESTORE VECTOR
 6438  004010  022626            2$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RETURN
 6439  004012  017737  175506  001126    MOV     @DRVCSR,$BDDAT  ;READ STATUS
 6440  004020  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;CORRECT?
 6441  004026  001401                    BEQ     3$              ;BR IF SO
 6442  004030  104005                    ERROR   5               ;INCORRECT STATUS ON READY INTR
 6443  004032  004737  010132    3$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
 6444
 6445                            ;;*******************************************************************
  (3)                            ;*TEST 14       TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
  (3)                            ;;*******************************************************************
  (2)  004036  000004            TST14:  SCOPE
  (2)
 6446  004040  106427  000200            MTPS    #200            ;DONT WANT ANY INTRS
 6447  004044  013737  001524  001122    MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6448  004052  005037  001124            CLR     $GDDAT          ;EXPECT 0
 6449  004056  012777  000001  175440    MOV     #1,@DRVCSR      ;SET GO WHICH SHOULD CLR READY
 6450  004064  017737  175434  001126    MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 6451  004072  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;CORRECT?
 6452  004100  001401                    BEQ     1$              ;BR IF SO
 6453  004102  104006                    ERROR   6               ;THE GO BIT FAILED TO CLR READY
 6454  004104  052777  000004  175412  1$:     BIS     #4,@DRVCSR      ;FNCT 2 SHOULD SET READY
 6455  004112  012737  002204  001124    MOV     #2204,$GDDAT    ;LD EXPECTED
 6456  004120  017737  175400  001126    MOV     @DRVCSR,$BDDAT  ;GET CSR
 6457  004126  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;CORRECT?
 6458  004134  001401                    BEQ     TST15           ;;NEXT TEST IF SET
 6459  004136  104006                    ERROR   6               ;FNCT 2 (VIA ATTN) FAILED TO SET READY
 6460
 6461                            ;;*******************************************************************
  (3)                            ;*TEST 15       TEST THAT READY CONTROLS 'BAR' BIT00
  (3)                            ;;*******************************************************************
  (2)  004140  000004            TST15:  SCOPE
  (2)
 6462  004142  012777  000004  175354    MOV     #4,@DRVCSR      ;SET READY
 6463  004150  013737  001522  001122    MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
 6464  004156  012737  000001  001124    MOV     #1,$GDDAT       ;EXPECT LSB OF BAR
 6465  004164  005077  175332            CLR     @DRVBAR         ;CLR BAR
 6466  004170  017737  175326  001126    MOV     @DRVBAR,$BDDAT  ;READ BAR
 6467  004176  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;CORRECT?
 6468  004204  001401                    BEQ     1$              ;BR IF SO
```

```
 6469  004206  104002                          ERROR   2               ;A00 FAILED TO READ A ONE(SB TIED TO RDY)
 6470  004210  012777  000001  175306  1$:     MOV     #1,@DRVCSR      ;SET GO(CLRS BAR BIT00)
 6471  004216  017737  175300  001126          MOV     @DRVBAR,$BDDAT  ;READ BAR
 6472  004224  001403                          BEQ     2$              ;BR IF ZERO
 6473  004226  005037  001124                  CLR     $GDDAT          ;EXPECTED ZERO
 6474  004232  104002                          ERROR   2               ;WHEN RDY CLRED-A00 FAILED TO READ A ZERO
 6475  004234  012777  000004  175262  2$:     MOV     #4,@DRVCSR      ;INSURE RDY SET BEFORE ADVANCING
 6476
 6477                                  ;;*******************************************************************
   (3)                                 ;*TEST 16       TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
   (3)                                 ;;*******************************************************************
   (2)  004242  000004                  TST16:  SCOPE
   (2)
 6478  004244  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 6479  004252  012737  125252  001124          MOV     #125252,$GDDAT  ;LD EXPECTED
 6480  004260  013777  001124  175240          MOV     $GDDAT,@DRVDBR  ;LD DBR WITH #125252
 6481  004266  012777  000400  175230          MOV     #400,@DRVCSR    ;SET CYCLE - SHOULD CLK DBR (IN)
 6482  004274  012777  052525  175224          MOV     #52525,@DRVDBR  ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
 6483  004302  017737  175220  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
 6484  004310  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6485  004316  001401                          BEQ     1$              ;BR IF SO
 6486  004320  104017                          ERROR   17              ;CYCLE DID NOT LATCH DBR (IN) DATA
 6487  004322  042777  000400  175174  1$:     BIC     #400,@DRVCSR    ;REMOVE CYCLE
 6488  004330  012737  052525  001124          MOV     #52525,$GDDAT   ;NOW EXPECT #52525
 6489  004336  017737  175164  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
 6490  004344  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6491  004352  001401                          BEQ     TST17           ;;NEXT TEST IF SO
 6492  004354  104002                          ERROR   2               ;DBR FAILED TO READ WHEN CYCLE CLRED (NORMAL)
 6493
 6494                                  ;;*******************************************************************
   (3)                                 ;*TEST 17       TEST SINGLE "DATI" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
   (3)                                 ;;*******************************************************************
   (2)  004356  000004                  TST17:  SCOPE
   (2)
 6495  004360  012737  004402  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6496  004366  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
 6497  004372  004476                          2$                      ;RETURN TO 2$ ON INTR
 6498  004374  012700  177776                  MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
 6499  004400  005001                          CLR     R1              ;R1 CONTROLS DATA SHIFTING
 6500  004402  012777  177777  175110  1$:     MOV     #-1,@DRVWCR     ;DO ONE XFER
 6501  004410  012777  015316  175104          MOV     #DBUF,@DRVBAR   ;GET DATA WORD FROM "DBUF"
 6502  004416  010037  015316                  MOV     R0,DBUF         ;SET UP MEM DATA
 6503  004422  012777  000101  175074          MOV     #101,@DRVCSR    ;SET IE & GO
 6504  004430  052777  000400  175066          BIS     #400,@DRVCSR    ;SET CYCLE
 6505  004436  106427  000000                  MTPS    #0              ;ENABLE THE INTR
 6506  004442  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6507  004450  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
 6508  004456  017737  175042  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 6509  004464  042777  000100  175032          BIC     #100,@DRVCSR    ;CLR IE
 6510  004472  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
 6511  004474  000442                          BR      4$              ;GO RESTORE VECTOR
 6512  004476  022626                  2$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
 6513  004500  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
 6514  004504  000700                          700                     ;CSR STATUS EXPECTED
 6515  004506  000001                          1                       ;# OF XFERS
 6516  004510  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
```

```
 6517  004512  000433                          BR      4$              ;GO RESTORE VECTOR
 6518  004514  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
 6519  004516  000431                          BR      4$              ;GO RESTORE VECTOR
 6520  004520  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
 6521  004522  000427                          BR      4$              ;GO RESTORE VECTOR
 6522  004524  012737  015316  001120          MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
 6523  004532  010037  001124                  MOV     R0,$GDDAT       ;LD EXPECTED
 6524  004536  017737  174764  001126          MOV     @DRVDBR,$BDDAT  ;READ DATA XFERED
 6525  004544  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6526  004552  001405                          BEQ     3$              ;BR IF SO
 6527  004554  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 6528  004562  104012                          ERROR   12              ;DATA ER - DBR CONTAINS WRONG DATA
 6529  004564  000406                          BR      4$              ;GO RESTORE VECTOR
 6530  004566  005100                  3$:     COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
 6531  004570  005101                          COM     R1              ;KEEP TRACK OF COMPLEMENT
 6532  004572  001303                          BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
 6533  004574  006300                          ASL     R0              ;WAS DONE - NOW SHIFT ZERO LEFT
 6534  004576  005200                          INC     R0              ;KEEP LSB SET
 6535  004600  103700                          BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
 6536  004602  004737  010132          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
 6537
 6538                                  ;;*******************************************************************
   (3)                                 ;*TEST 20       TEST SINGLE "DATO" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
   (3)                                 ;;*******************************************************************
   (2)  004606  000004                  TST20:  SCOPE
   (2)
 6539  004610  012737  004632  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6540  004616  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
 6541  004622  004720                          2$                      ;RETURN TO 2$ ON INTR
 6542  004624  012700  177776                  MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
 6543  004630  005001                          CLR     R1              ;R1 CONTROLS DATA SHIFTING
 6544  004632  012777  177777  174660  1$:     MOV     #-1,@DRVWCR     ;DO ONE XFER
 6545  004640  012777  015316  174654          MOV     #DBUF,@DRVBAR   ;WRITE DATA WORD TO "DBUF"
 6546  004646  010077  174654                  MOV     R0,@DRVDBR      ;SET UP DATA IN DBR
 6547  004652  012777  000103  174644          MOV     #103,@DRVCSR    ;SET IE, GO & FNCT1 (C1 CONTROL)
 6548  004660  052777  000400  174636          BIS     #400,@DRVCSR    ;SET CYCLE
 6549  004666  106427  000000                  MTPS    #0              ;ENABLE THE INTR
 6550  004672  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6551  004700  012737  001702  001124          MOV     #1702,$GDDAT    ;LD EXPECTED
 6552  004706  017737  174612  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 6553  004714  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
 6554  004716  000442                          BR      4$              ;GO RESTORE VECTOR
 6555  004720  022626                  2$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
 6556  004722  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
 6557  004726  001702                          1702                    ;CSR STATUS EXPECTED
 6558  004730  000001                          1                       ;# OF XFERS
 6559  004732  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
 6560                                                                  ;CYCLE, READY, IE & FNCT 1
 6561  004734  000433                          BR      4$              ;GO RESTORE VECTOR
 6562  004736  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
 6563  004740  000431                          BR      4$              ;GO RESTORE VECTOR
 6564  004742  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
 6565  004744  000427                          BR      4$              ;GO RESTORE VECTOR
 6566  004746  012737  015316  001120          MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
 6567  004754  010037  001124                  MOV     R0,$GDDAT       ;LD EXPECTED
 6568  004760  013737  015316  001126          MOV     DBUF,$BDDAT     ;GET DATA XFERED
```

```
  6569  004766  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
  6570  004774  001405                          BEQ     3$              ;BR IF SO
  6571  004776  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
  6572  005004  104013                          ERROR   13              ;DATA ER - MEM CONTAINS WRONG DATA
  6573  005006  000406                          BR      4$              ;GO RESTORE VECTOR
  6574  005010  005100                  3$:     COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
  6575  005012  005101                          COM     R1              ;KEEP TRACK OF COMPLEMENT
  6576  005014  001306                          BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
  6577  005016  006300                          ASL     R0              ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
  6578  005020  005200                          INC     R0              ;KEEP LSB SET
  6579  005022  103703                          BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
  6580  005024  004737  010132          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
  6581
  6582                                  ;;*******************************************************************
   (3)                                  ;*TEST 21       TEST 200 "DATI" NPR TRANSFERS (BURST MODE)
   (3)                                  ;;*******************************************************************
   (2)  005030  000004                  TST21:  SCOPE
   (2)
   (1)  005032  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  6583  005040  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
  6584  005044  005142                          1$                      ;RETURN TO 1$ ON INTR
  6585  005046  004737  010336                  JSR     PC,LDBUF        ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
  6586  005052  012777  177470  174440          MOV     #-200.,@DRVWCR  ;LOAD WC REG - WILL DO 200 XFERS
  6587  005060  012777  015316  174434          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
  6588  005066  106427  000000                  MTPS    #0              ;ENABLE THE INTR
  6589  005072  012777  000101  174424          MOV     #101,@DRVCSR    ;SET IE & GO
  6590  005100  052777  000400  174416          BIS     #400,@DRVCSR    ;SET CYCLE
  6591  005106  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
  6592  005114  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
  6593  005122  017737  174376  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
  6594  005130  042777  000100  174366          BIC     #100,@DRVCSR    ;CLR INTR ENABLE
  6595  005136  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
  6596  005140  000434                          BR      2$              ;GO RESTORE VECTOR
  6597  005142  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
  6598  005144  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
  6599  005150  000700                          700                     ;CSR STATUS EXPECTED
  6600  005152  000310                          200.                    ;# OF XFERS
  6601  005154  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
  6602  005156  000425                          BR      2$              ;GO RESTORE VECTOR
  6603  005160  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
  6604  005162  000423                          BR      2$              ;GO RESTORE VECTOR
  6605  005164  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
  6606  005166  000421                          BR      2$              ;GO RESTORE VECTOR
  6607  005170  012737  016134  001120          MOV     #DBUF+616,$GDADR  ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
  6608  005176  012737  070707  001124          MOV     #70707,$GDDAT   ;LD EXPECTED
  6609  005204  017737  174316  001126          MOV     @DRVDBR,$BDDAT  ;DBR SHOULD HAVE LAST DATUM
  6610  005212  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
  6611  005220  001404                          BEQ     2$              ;BR IF SO
  6612  005222  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
  6613  005230  104012                          ERROR   12              ;DATA ER - DBR DID NOT CONTAIN EXPECTED LAST XFER
  6614  005232  004737  010132          2$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
  6615
  6616                                  ;;*******************************************************************
   (3)                                  ;*TEST 22       TEST 200 "DATO" NPR TRANSFERS (BURST MODE)
   (3)                                  ;;*******************************************************************
   (2)  005236  000004                  TST22:  SCOPE
```

```
   (2)
   (1)  005240  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  6617  005246  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
  6618  005252  005352                          1$                      ;RETURN TO 1$ ON INTR
  6619  005254  012777  177470  174236          MOV     #-200.,@DRVWCR  ;WORD WC REG - WILL DO 200 XFER'S
  6620  005262  012777  015316  174232          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
  6621  005270  012777  177377  174230          MOV     #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
  6622  005276  106427  000000                  MTPS    #0              ;ENABLE THE INTR
  6623  005302  012777  000103  174214          MOV     #103,@DRVCSR    ;SET IE, FNCT 1 & GO
  6624  005310  052777  000400  174206          BIS     #400,@DRVCSR    ;SET CYCLE
  6625  005316  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
  6626  005324  012737  001702  001124          MOV     #1702,$GDDAT    ;LD EXPECTED
  6627  005332  017737  174166  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
  6628  005340  042777  000100  174156          BIC     #100,@DRVCSR    ;CLR INTR ENABLE
  6629  005346  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
  6630  005350  000416                          BR      2$              ;GO RESTORE VECTOR
  6631  005352  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
  6632  005354  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
  6633  005360  001702                          1702                    ;CSR STATUS EXPECTED
  6634  005362  000310                          200.                    ;# OF XFERS
  6635  005364  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
  6636                                                                  ;CYCLE, READY, IE & FNCT 1
  6637  005366  000407                          BR      2$              ;GO RESTORE VECTOR
  6638  005370  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
  6639  005372  000405                          BR      2$              ;GO RESTORE VECTOR
  6640  005374  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
  6641  005376  000403                          BR      2$              ;GO RESTORE VECTOR
  6642  005400  004737  010460                  JSR     PC,CKDAT        ;RETURN HERE IF OK - NOW GO CHECK DATA
  6643  005404  104013                          ERROR   13              ;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
  6644  005406  004737  010132          2$:     JSR     PC,RSTVEC       ;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
  6645
  6646                                  ;;*****************************************************************
   (3)                                  ;*TEST 23       TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
   (3)                                  ;;*****************************************************************
   (2)  005412  000004                  TST23:  SCOPE
   (2)
   (1)  005414  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  6647  005422  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
  6648  005430  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
  6649  005434  005544                          3$                      ;RETURN TO 3$ ON INTR
  6650  005436  012700  000010                  MOV     #10,R0          ;DO EIGHT 200 WORD XFER'S
  6651  005442  005037  001126                  CLR     $BDDAT          ;USR $BDDAT AS A COUNTER
  6652  005446  012777  177470  174044  1$:     MOV     #-200.,@DRVWCR  ;DO 200 XFERS (DATI'S)
  6653  005454  012777  015316  174040          MOV     #DBUF,@DRVBAR   ;FROM DBUF
  6654  005462  106427  000000                  MTPS    #0              ;ALLOW AN INTR
  6655  005466  012777  000101  174030          MOV     #101,@DRVCSR    ;SET IE & GO
  6656  005474  052777  000400  174022          BIS     #400,@DRVCSR    ;SET CYCLE
  6657  005502  000240                          NOP                     ;FREEBEE
  6658  005504  000240                          NOP
  6659  005506  000240                          NOP
  6660  005510  005237  001126          2$:     INC     $BDDAT          ;START COUNTING - SHOULD NEVER GET HERE
  6661  005514  001375                          BNE     2$              ;UNTIL 64K
  6662  005516  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
  6663  005524  017737  173774  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
  6664  005532  042777  000100  173764          BIC     #100,@DRVCSR    ;CLR IE
  6665  005540  104005                          ERROR   5               ;NO INTERRUPT ON 200 DATI'S
```

```
 6666 005542 000407                           BR      4$                ;GO RESTORE VECTOR
 6667 005544 022626                   3$:     CMP     (SP)+,(SP)+       ;FIX STACK SINCE NO RTI
 6668 005546 005300                           DEC     R0                ;DONE 8 TIMES?
 6669 005550 001336                           BNE     1$                ;BR IF NOT
 6670 005552 005737 001126                    TST     $BDDAT            ;SHOULD STILL BE ZERO
 6671 005556 001401                           BEQ     4$                ;BR IF SO
 6672 005560 104014                           ERROR   14                ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
 6673 005562 004737 010132            4$:     JSR     PC,RSTVEC         ;GO RESTORE VECTOR
 6674
 6675                                 ;;*********************************************************************
  (3)                                 ;*TEST 24        TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
  (3)                                 ;;*********************************************************************
  (2) 005566 000004                   TST24:  SCOPE
  (2)
  (1) 005570 012737 000010 001160             MOV     #10,$TIMES        ;;DO 10 ITERATIONS
 6676 005576 013737 001524 001122             MOV     DRVCSR,$BDADR     ;SET UP CSR ADRS
 6677 005604 004537 010112                    JSR     R5,SETVEC         ;GO SET UP INTR RETURN
 6678 005610 005716                                   3$                ;RETURN TO 3$ ON INTR
 6679 005612 012700 000010                    MOV     #10,R0            ;DO EIGHT 200 WORD XFER'S
 6680 005616 012737 000000 001126             MOV     #0,$BDDAT         ;USE $BDDAT AS A COUNTER
 6681 005624 012777 177470 173666     1$:     MOV     #-200.,@DRVWCR    ;DO 200 XFERS (DATI'S)
 6682 005632 012777 015316 173662             MOV     #DBUF,@DRVBAR     ;FROM DBUF
 6683 005640 106427 000000                    MTPS    #0                ;ALLOW AN INTR
 6684 005644 012777 000111 173652             MOV     #111,@DRVCSR      ;SET IE, FNCT3 & GO
 6685 005652 052777 000400 173644             BIS     #400,@DRVCSR      ;SET CYCLE
 6686 005660 000240                           NOP                       ;FREEBEE
 6687 005662 005237 001126            2$:     INC     $BDDAT            ;START COUNTING
 6688 005666 001375                           BNE     2$                ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
 6689 005670 012737 004710 001124             MOV     #4710,$GDDAT      ;LD EXPECTED
 6690 005676 017737 173622 001126             MOV     @DRVCSR,$BDDAT    ;READ STATUS
 6691 005704 042777 000100 173612             BIC     #100,@DRVCSR      ;CLR IE
 6692 005712 104005                           ERROR   5                 ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
 6693 005714 000423                           BR      5$                ;GO RESTORE VECTOR
 6694 005716 022626                   3$:     CMP     (SP)+,(SP)+       ;FIX STACK SINCE NO RTI
 6695 005720 005300                           DEC     R0                ;DONE 8 TIMES?
 6696 005722 001340                           BNE     1$                ;BR IF NOT
 6697 005724 022737 000000 001126             CMP     #0,$BDDAT         ;$BDDAT SHOULD HAVE BEEN COUNTED
 6698 005732 103401                           BCS     4$                ;BR IF SO
 6699 005734 104015                           ERROR   15                ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
 6700 005736 017737 173562 001126     4$:     MOV     @DRVCSR,$BDDAT    ;READ STATUS
 6701 005744 012737 004710 001124             MOV     #4710,$GDDAT      ;LD EXPECTED
 6702 005752 023737 001124 001126             CMP     $GDDAT,$BDDAT     ;CORRECT?
 6703 005760 001401                           BEQ     5$                ;BR IF SO
 6704 005762 104007                           ERROR   7                 ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
 6705 005764 004737 010132            5$:     JSR     PC,RSTVEC         ;GO RESTORE VECTOR
 6706
 6707                                 ;;*********************************************************************
  (3)                                 ;*TEST 25        TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
  (3)                                 ;;*********************************************************************
  (2) 005770 000004                   TST25:  SCOPE
  (2)
  (1) 005772 012737 000200 001160             MOV     #200,$TIMES       ;;DO 200 ITERATIONS
 6708 006000 004537 010112                    JSR     R5,SETVEC         ;GO SET UP INTR RETURN
 6709 006004 006122                                   2$                ;RETURN TO 2$ ON INTR
 6710 006006 004737 010410                    JSR     PC,LDBUF1         ;GO SET UP DBUF (SPECIAL COM PATTERN)
 6711 006012 012737 011702 006130             MOV     #11702,3$         ;3$ CONTAINS EXPECTED STATUS
```

```
6712  006020  012737  000001  006132          MOV     #1,4$           ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
6713  006026  106427  000000          1$:     MTPS    #0              ;ALLOW INTR
6714  006032  012777  015316  173462          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
6715  006040  013777  006132  173452          MOV     4$,@DRVWCR      ;GET XFER #
6716  006046  005477  173446                  NEG     @DRVWCR         ;NEGATE FOR WC
6717  006052  012777  010101  173444          MOV     #10101,@DRVCSR  ;SET UP MAINT, IE & GO
6718  006060  052777  000400  173436          BIS     #400,@DRVCSR    ;SET CYCLE
6719  006066  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
6720  006074  013737  006130  001124          MOV     3$,$GDDAT       ;LD EXPECTED
6721  006102  017737  173416  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
6722  006110  042777  000100  173406          BIC     #100,@DRVCSR    ;DISABLE IE
6723  006116  104005                          ERROR   5               ;NO INTR ON XFER (IN MAINT MD)
6724  006120  000440                          BR      6$              ;GO RESTORE VECTOR
6725  006122  022626                  2$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
6726  006124  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
6727  006130  011702                  3$:     11702                   ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
6728  006132  000001                  4$:     1                       ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
6729  006134  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - WILL EXPECT
6730                                                                  ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
6731                                                                  ;CYCLE, READY & IE
6732  006136  000431                          BR      6$              ;GO RESTORE VECTOR
6733  006140  104010                          ERROR   10              ;RETURN HERE IF WC ER - SHOULD BE 0
6734  006142  000427                          BR      6$              ;GO RESTORE VECTOR
6735  006144  104011                          ERROR   11              ;RETURN HERE IF BAR ER-
6736  006146  000425                          BR      6$              ;GO RESTORE VECTOR
6737  006150  062737  001002  006130          ADD     #1002,3$        ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
6738  006156  032737  020000  006130          BIT     #BIT13,3$       ;LOOK FOR OVERFLOW
6739  006164  001403                          BEQ     5$              ;BR IF NOT
6740  006166  012737  010700  006130          MOV     #10700,3$       ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
6741  006174  005237  006132          5$:     INC     4$              ;ADVANCE CURRENT XFER #
6742  006200  022737  000011  006132          CMP     #11,4$          ;HAVE 10 XFERS BEEN DONE
6743  006206  001307                          BNE     1$              ;BR IF NOT
6744  006210  004537  010526                  JSR     R5,CKDAT1       ;NOW GO CHECK DATA
6745  006214  000010                          10                      ;# OF XFER'S TO CHECK
6746  006216  104013                          ERROR   13              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
6747  006220  000240                          NOP                     ;RESTORE VECTOR NEXT
6748  006222  004737  010132          6$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
6749
6750                                  ;;*******************************************************************
  (3)                                 ;*TEST 26       TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
  (3)                                 ;;*******************************************************************
  (2)   006226  000004                TST26:  SCOPE
  (2)
  (1)   006230  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
6751    006236  012700  000002                  MOV     #2,R0           ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
6752    006242  004537  010112          1$:     JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
6753    006246  006340                          2$                      ;RETURN TO 2$ ON TIMEOUT INTR
6754    006250  012777  177777  173242          MOV     #-1,@DRVWCR     ;SET UP FOR ONE XFER'S
6755    006256  012777  160000  173236          MOV     #160000,@DRVBAR ;SET UP CA TO 160000 (RESERVED)
6756    006264  106427  000000                  MTPS    #0              ;ALLOW INTR
6757    006270  012777  000161  173226          MOV     #161,@DRVCSR    ;SET IE, XAD 17,16 & GO
6758    006276  052777  000400  173220          BIS     #400,@DRVCSR    ;SET CYCLE
6759    006304  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS - SHOULD NEVER GET HERE
6760    006312  012737  140760  001124          MOV     #140760,$GDDAT  ;LD EXPECTED
6761    006320  017737  173200  001126          MOV     @DRVCSR,$BDDAT  ;GIVE THEM THE STATUS
6762    006326  042777  000100  173170          BIC     #100,@DRVCSR    ;CLR IE
```

```
                                        ERROR   16              ;NEX FAILED TO CAUSE AN INTERRUPT
                                        BR      7$              ;GO RESTORE VECTOR
                                2$:     CMP     (SP)+,(SP)+     ;SHOULD INTR RETURN HERE - FIX STACK
006342  017737  173156  001126        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
006350  012737  140760  001124        MOV     #140760,$GDDAT  ;LD EXPECTED
006356  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
006364  001405                        BEQ     3$              ;BR IF SO
006366  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
006374  104016                        ERROR   16              ;STATUS ER - EXPECTED
                                                              ;ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
006376  000501                        BR      7$              ;GO RESTORE VECTOR
006400  017737  173114  001126  3$:   MOV     @DRVWCR,$BDDAT  ;READ WORD COUNT
006406  012737  177777  001124        MOV     #-1,$GDDAT      ;SHOULD STILL HAVE -1
006414  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
006422  001405                        BEQ     4$              ;BR IF SO
006424  013737  001520  001122        MOV     DRVWCR,$BDADR   ;SET UP WCR ADRS
006432  104010                        ERROR   10              ;WC INCREMENTED ON A TIMEOUT ER
006434  000462                        BR      7$              ;GO RESTORE VECTOR
006436  017737  173060  001126  4$:   MOV     @DRVBAR,$BDDAT  ;READ BUFFER ADRS
006444  012737  160000  001124        MOV     #160000,$GDDAT  ;SHOULD NOT HAVE INCREMENTED
006452  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
006460  001405                        BEQ     5$              ;BR IF SO
006462  013737  001522  001122        MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
006470  104011                        ERROR   11              ;BAR INCREMENTED ON A TIMEOUT ER
006472  000443                        BR      7$              ;GO RESTORE VECTOR
006474  005300                5$:     DEC     R0              ;KEEP TRACK ON HOW TO CLR
006476  001422                        BEQ     6$              ;BR IF CLR BY RESET
006500  042777  040000  173016        BIC     #40000,@DRVCSR  ;WRITE NEX TO ZERO
006506  017737  173012  001126        MOV     @DRVCSR,$BDDAT  ;READ CSR
006514  012737  000760  001124        MOV     #760,$GDDAT     ;LD EXPECTED
006522  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
006530  001644                        BEQ     1$              ;BR IF SO + REPEAT TEST FOR RESET TEST
006532  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
006540  104016                        ERROR   16              ;'NEX' FAILED TO WRITE TO ZERO
006542  000417                        BR      7$              ;GO RESTORE VECTOR
006544  000005                6$:     RESET                   ;ISSUE BUS RESET
006546  017737  172752  001126        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
006554  012737  000200  001124        MOV     #200,$GDDAT     ;EXPECT ONLY READY
006562  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
006570  001404                        BEQ     7$              ;BR IF SO
006572  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
006600  104016                        ERROR   16              ;RESET FAILED TO CLR 'NEX'
006602  004737  010132        7$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
                                ;;**************************************************************
                                ;*TEST 27        TEST 200 NPR TRANSFERS IN MAINT MODE
                                ;;**************************************************************
006606  000004                TST27:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
006616  004537  010112                JSR     R5,SETVEC       ;GO SET UP INTR RETURN
006622  006734                        2$                      ;RETURN TO 2$ ON INTR
006624  004737  010410                JSR     PC,LDBUF1       ;GO SET UP DBUF (SPECIAL COM PATTERN)
006630  012777  015316  172664        MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
006636  012777  177470  172654        MOV     #-200.,@DRVWCR  ;SET UP FOR 200 XFER'S
006644  106427  000000                MTPS    #0              ;ALLOW INTR
006650  012777  010101  172646        MOV     #10101,@DRVCSR  ;SET MAINT, IE & GO
```

```
     006656 052777 000400 172640          BIS    #400,@DRVCSR      ;SET CYCLE
     006664 012737 000000 001126          MOV    #0,$BDDAT         ;SET UP A COUNTER
     006672 005237 001126          1$:    INC    $BDDAT            ;COUNT AWAY
     006676 001375                        BNE    1$                ;WAIT TILL DONE - SHOULD INTR BEFORE OVFLO
     006700 013737 001524 001122          MOV    DRVCSR,$BDADR     ;SET UP CSR ADRS
     006706 012737 010700 001124          MOV    #10700,$GDDAT     ;LD EXPECTED
     006714 017737 172604 001126          MOV    @DRVCSR,$BDDAT    ;READ STATUS
     006722 042777 000100 172574          BIC    #100,@DRVCSR      ;DISABLE IE
     006730 104005                        ERROR  5                 ;NO INTR AFTER 200 MAINT MODE XFER'S
     006732 000420                        BR     3$                ;GO RESTORE VECTOR
     006734 022626                 2$:    CMP    (SP)+,(SP)+       ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
     006736 004537 010156                 JSR    R5,CKSTAT         ;GO CHECK STATUS
     006742 010700                        10700                    ;EXPECTED STATUS
     006744 000310                        200.                     ;# OF XFERS
     006746 104007                        ERROR  7                 ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
                                                                   ;CYCLE, READY & IE
     006750 000411                        BR     3$                ;GO RESTORE VECTOR
     006752 104010                        ERROR  10                ;RETURN HERE IF WC ER - SHOULD BE 0
     006754 000407                        BR     3$                ;GO RESTORE VECTOR
     006756 104011                        ERROR  11                ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
     006760 000405                        BR     3$                ;GO RESTORE VECTOR
     006762 004537 010526                 JSR    R5,CKDAT1         ;RETURN HERE IF OK - NOW GO CHECK DATA
     006766 000310                        200.                     ;# OF XFER'S TO CHECK
     006770 104013                        ERROR  13                ;RETURN HERE IF DATA ER - (WITH MAINT SET)
     006772 000240                        NOP                      ;RESTORE VECTOR NEXT
     006774 004737 010132          3$:    JSR    PC,RSTVEC         ;GO RESTORE VECTOR
                                   ;;*****************************************************************
 (3)                               ;*TEST 30       TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
 (3)                               ;;*****************************************************************
 (2) 007000 000004                 TST30: SCOPE
 (2)
 (1) 007002 012737 000005 001160          MOV    #5,$TIMES         ;;DO 5 ITERATIONS
     007010 004537 010112                 JSR    R5,SETVEC         ;GO SET UP INTR RETURN
     007014 007150                        3$                       ;RETURN TO 3$ ON INTR
     007016 005037 001542                 CLR    DBUFP             ;GET LOWEST BUFFER ADRS
     007022 062737 020000 001542   1$:    ADD    #20000,DBUFP      ;POINT TO NEXT 4K
     007030 023737 001540 001542          CMP    CORSZ,DBUFP       ;IS THE 4K THERE?
     007036 001465                        BEQ    4$                ;BR IF NOT
     007040 004737 010410                 JSR    PC,LDBUF1         ;GO SET UP SPECIAL COMPEMENT PATTERN
     007044 013777 001542 172450          MOV    DBUFP,@DRVBAR     ;SET UP BUFFER ADRS
     007052 012777 177470 172440          MOV    #-200.,@DRVWCR    ;SET UP FOR 200 XFER'S
     007060 106427 000000                 MTPS   #0                ;ALLOW INTR
     007064 012777 010101 172432          MOV    #10101,@DRVCSR    ;SET MAINT,IE & GO
     007072 052777 000400 172424          BIS    #400,@DRVCSR      ;SET CYCLE
     007100 012737 000000 001126          MOV    #0,$BDDAT         ;SET UP A COUNTER
     007106 005237 001126          2$:    INC    $BDDAT            ;COUNT AWAY
     007112 001375                        BNE    2$                ;SHOULD ALWAYS INTR FROM THIS LOOP
     007114 013737 001524 001122          MOV    DRVCSR,$BDADR     ;SET UP CSR ADRA
     007122 012737 010700 001124          MOV    #10700,$GDDAT     ;LD EXPECTED
     007130 017737 172370 001126          MOV    @DRVCSR,$BDDAT    ;READ CSR
     007136 042777 000100 172360          BIC    #100,@DRVCSR      ;DISABLE IE
     007144 104005                        ERROR  5                 ;NO INTR AFTER 200 MAINT MODE XFER'S
     007146 000421                        BR     4$                ;GO RESTORE VECTOR
     007150 022626                 3$:    CMP    (SP)+,(SP)+       ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
     007152 004537 010156                 JSR    R5,CKSTAT         ;GO CHECK STATUS
     007156 010700                        10700                    ;EXPECTED STATUS
```

```
 6865  007160  000310                          200.                    ;# OF XFER'S
 6866  007162  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
 6867                                                                  ;CYCLE, READY & IE
 6868  007164  000412                          BR      4$              ;GO RESTORE VECTOR
 6869  007166  104010                          ERROR   10              ;RETURN HERE IF WC ER - SHOULD = 0
 6870  007170  000410                          BR      4$              ;GO RESTORE VECTOR
 6871  007172  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUFP+620
 6872  007174  000406                          BR      4$              ;GO RESTORE VECTOR
 6873  007176  004537  010526                  JSR     R5,CKDAT1       ;RETURN HERE IF OK - NOW GO CK DATA
 6874  007202  000310                          200.                    ;# OF XFER'S TO CK
 6875  007204  104013                          ERROR   13              ;RETURN HERE IF DATA ER
 6876  007206  000401                          BR      4$              ;GO RESTORE VECTOR
 6877  007210  000704                          BR      1$              ;TRY NEXT BANK
 6878  007212  012737  015316  001542  4$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
 6879  007220  004737  010132                  JSR     PC,RSTVEC       ;GO RESTORE VECTOR
 6880                                  ;;*******************************************************************
  (3)                                  ;*TEST 31       TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
  (3)                                  ;;*******************************************************************
  (2)  007224  000004                  TST31:  SCOPE
  (2)
 6881  007226  105777  171706                  TSTB    @SWR            ;KDF11-B?
 6882  007232  100507                          BMI     ALTERN          ;IF YES,GO TO THE NEXT SUBTEST
 6883  007234  004537  010112          101$:   JSR     R5,SETVEC       ;GO SET UP INTR RETURN
 6884  007240  007344                          1$                      ;RETURN TO 1$ ON INTR
 6885  007242  012777  177777  172250          MOV     #-1,@DRVWCR     ;SET UP WC - 1 XFER
 6886  007250  013737  001150  001542          MOV     $TPS,DBUFP      ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
 6887  007256  013777  001150  172236          MOV     $TPS,@DRVBAR    ;SET UP BUFFER ADRS - FROM PRINTER CSR
 6888  007264  106427  000000                  MTPS    #0              ;ALLOW INTR
 6889  007270  005077  172232                  CLR     @DRVDBR         ;ZERO THE DBR
 6890  007274  012777  000161  172222          MOV     #161,@DRVCSR    ;SET IE, XAD17 & XAD16, & GO
 6891  007302  052777  000400  172214          BIS     #400,@DRVCSR    ;SET CYCLE
 6892  007310  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6893  007316  012737  000760  001124          MOV     #760,$GDDAT     ;LD EXPECTED STATUS
 6894  007324  017737  172174  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 6895  007332  042777  000100  172164          BIC     #100,@DRVCSR    ;DISABLE IE
 6896  007340  104005                          ERROR   5               ;NO INTR ON 1 WD XFER FROM XCSR
 6897  007342  000434                          BR      2$              ;GO RESTORE VECTOR
 6898  007344  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STK SINCE NO RTI
 6899  007346  004537  010156                  JSR     R5,CKSTAT       ;GO CK STATUS
 6900  007352  000760                          760                     ;CSR EXPECTED STATUS
 6901  007354  000001                          1                       ;# OF XFER'S
 6902  007356  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
 6903                                                                  ;XAD17 & XAD16, RDY & IE
 6904  007360  000425                          BR      2$              ;GO RESTORE VECTOR
 6905  007362  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
 6906  007364  000423                          BR      2$              ;GO RESTORE VECTOR
 6907  007366  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
 6908  007370  000421                          BR      2$              ;GO RESTORE VECTOR
 6909  007372  017737  171552  001124          MOV     @$TPS,$GDDAT    ;RETURN HERE IF OK - GET XCSR CONTENTS
 6910  007400  017737  172122  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
 6911  007406  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6912  007414  001407                          BEQ     2$              ;BR IF SO
 6913  007416  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 6914  007424  013737  001150  001120          MOV     $TPS,$GDADR     ;GET ADRS OF DATA (XCSR)
 6915  007432  104020                          ERROR   20              ;DATA ER FROM TTY PRINTER CSR
 6916  007434  012737  015316  001542  2$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
```

```
                                        JSR     PC,RSTVEC       ;GO RESTORE VECTOR
                                        JMP     NXDEV           ;IGNORE NEXT SUBTEST

                                ;;**************************************************************
                                ;IN CASE OF KDF11-B TEST SOME SPECIFIED I/O PAGE
                                ;;**************************************************************
                                ALTERN: TST     IOPAGE          ;I/O TRANSFER DESIRED?
                                        BEQ     NXDEV           ;IF NOT, SKIP THE TEST
                                        JSR     R5,SETVEC       ;GO SET UP INTR RETURN
                                        1$                      ;RETURN TO 1$ ON INTR
                                        MOV     #-1,@DRVWCR     ;SET UP WC = 1 XFER
                                        MOV     IOPAGE,DBUFP    ;SET UP BUFFER ADRS TO I/O CSR ADRS
                                        MOV     IOPAGE,@DRVBAR  ;SET UP BUFFER ADRS - FROM I/O CSR
                                        MTPS    #0              ;ALLOW INTR
                                        CLR     @DRVDBR         ;ZERO THE DBR
                                        MOV     #161,@DRVCSR    ;SET IE, XAD17 & XAD16, & GO
                                        BIS     #400,@DRVCSR    ;SET CYCLE
                                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                        MOV     #760,$GDDAT     ;LD EXPECTED STATUS
                                        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
                                        BIC     #100,@DRVCSR    ;DISABLE IE
                                        ERROR   5               ;NO INTR ON 1 WD XFER FROM XCSR
                                        BR      2$              ;GO RESTORE VECTOR
                                1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STK SINCE NO RTI
                                        JSR     R5,CKSTAT       ;GO CK STATUS
                                        760                     ;CSR EXPECTED STATUS
                                        1                       ;# OF XFER'S
                                        ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
                                                                ;XAD17 & XAD16, RDY & IE
                                        BR      2$              ;GO RESTORE VECTOR
                                        ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
                                        BR      2$              ;GO RESTORE VECTOR
                                        ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
                                        BR      2$              ;GO RESTORE VECTOR
                                        MOV     @IOPAGE,$GDDAT  ;RETURN HERE IF OK - GET XCSR CONTENTS
                                        MOV     @DRVDBR,$BDDAT  ;READ DBR
                                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                                        BEQ     2$              ;BR IF SO
                                        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
                                        MOV     IOPAGE,$GDADR   ;GET ADRS OF DATA (XCSR)
                                        ERROR   20              ;DATA ER FROM TTY PRINTER CSR
                                2$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
                                        JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                                ;;**************************************************************
                                ;DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
                                ;;**************************************************************
                                NXDEV:  SCOPE
                                NXDEV1: CLC                     ;CLR CARRY
                                        ROR     DMAP            ;LOOK FOR NEXT
                                        BEQ     $EOP            ;BR IF ALL TESTED
                                        ADD     #10,DRVWCR      ;OFFSET BASE BUS ADRS TO NEXT DRV11B
                                        ADD     #10,DRVBAR      ;
                                        ADD     #10,DRVCSR      ;
                                        ADD     #10,DRVDBR      ;
                                        ADD     #4,DRVCTO       ;OFFSET VECTOR ADRS TO NEXT
```

```
  6973  007742  062737  000004  001532          ADD     #4,DRVCT2       ;
  6974  007750  005237  001206                  INC     $UNIT           ;COUNT DEVICE
  6975  007754  032737  000001  001536          BIT     #1,DMAP         ;IS IT SELECTED?
  6976  007762  001744                          BEQ     NXDEV1          ;BR IF NOT
  6977  007764  000137  002406                  JMP     RESTRT          ;TEST NEXT
  6978                                .SBTTL  END OF PASS ROUTINE
   (1)
   (2)                                ;;**********************************************************************
   (1)                                ;*INCREMENT THE PASS NUMBER ($PASS)
   (1)                                ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
   (1)                                ;*IF THERES A MONITOR GO TO IT
   (1)                                ;*IF THERE ISN'T JUMP TO START1
   (1)
   (1)  007770                        $EOP:
   (2)  007770  000240                        NOP
   (1)  007772  005037  001102                CLR     $TSTNM          ;;ZERO THE TEST NUMBER
   (1)  007776  005037  001160                CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
   (1)  010002  005237  001202                INC     $PASS           ;;INCREMENT THE PASS NUMBER
   (1)  010006  042737  100000  001202        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
   (1)  010014  005327                        DEC     (PC)+           ;;LOOP?
   (1)  010016  000001                $EOPCT: .WORD   1
   (1)  010020  003022                        BGT     $DOAGN          ;;YES
   (1)  010022  012737                        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
   (1)  010024  000001                $ENDCT: .WORD   1
   (1)  010026  010016                        $EOPCT
   (1)  010030  104401  010075                TYPE    ,$ENDMG         ;;TYPE "END PASS #"
   (2)  010034  013746  001202                MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
   (2)  010040  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
   (1)  010042  104401  010072                TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
   (1)  010046  013700  000042        $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
   (1)  010052  001405                        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
   (1)  010054  000005                        RESET                   ;;CLEAR THE WORLD
   (1)  010056  004710                $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
   (1)  010060  000240                        NOP                     ;;SAVE ROOM
   (1)  010062  000240                        NOP                     ;;FOR
   (1)  010064  000240                        NOP                     ;;ACT11
   (1)  010066                        $DOAGN:
   (1)  010066  000137                        JMP     @(PC)+          ;;RETURN
   (1)  010070  002012                $RTNAD: .WORD   START1
   (1)
   (1)
   (1)  010072     377     377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
   (1)  010075     015  042412  042116 $ENDMG: .ASCIZ  <15><12>/END PASS #/
   (1)  010102  050040  051501  020123
   (1)  010110  000043
```

```
                                .SBTTL  PROGRAM SUBROUTINES


                                ;;*****************************************************************
                                ;THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
                                ;SETS UP THE DRV11B INTERRUPT TO RETURN ON INTERRUPT
                                ;TO THE ADDRESS INDICATED ((R5)) BY THE CALL +2
                                ;;*****************************************************************
010112  106427  000200          SETVEC: MTPS    #200            ;SET UP FOR NO INTERRUPT
010116  012577  171406                  MOV     (R5)+,@DRVCT0   ;SET UP INTR RETURN ADRS
010122  012777  000200  171402          MOV     #200,@DRVCT2    ;KEEP PRIORITY LEVEL AT TOP ON INTR
010130  000205                          RTS     R5              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE CLEARS THE DRV11B CSR - RESTORES THE DRV11B
                                ;INTERRUPT VECTOR TO A HALT - RAISES PRIORITY LEVEL
                                ;;*****************************************************************
010132  005077  171366          RSTVEC: CLR     @DRVCSR         ;CLR STATUS & CONTROL
010136  013777  001532  171364          MOV     DRVCT2,@DRVCT0  ;POINT VECTOR TO HALT
010144  005077  171362                  CLR     @DRVCT2         ;SET UP HALT
010150  106427  000200                  MTPS    #200            ;RAISE PRIORITY LEVEL
010154  000207                          RTS     PC              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE CHECKS ON ALL DATA TRANSFERS FOR: CORRECT STATUS,
                                ;CORRECT WORD COUNT & CORRECT BUFFER ADDRESS - THE EXPECTED DATA IS
                                ;SUPPLIED IN THE CALL +2 & +4 - THE RETURN IS TO +22 IF NO ERRORS
                                ;DETECTED - IF AN ERROR IS DETECTED THE RETURN IS TO THE APPROPRIATE ERROR EMT
                                ;;*****************************************************************
010156  017737  171342  001126  CKSTAT: MOV     @DRVCSR,$BDDAT  ;READ THE STATUS
010164  042777  000100  171332          BIC     #100,@DRVCSR    ;DISABLE THE IE BIT
010172  012537  001124                  MOV     (R5)+,$GDDAT    ;SET UP EXPECTED STATUS
010176  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
010204  001406                          BEQ     1$              ;BR IF SO
010206  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
010214  062705  000002                  ADD     #2,R5           ;POINT TO THE CSR ER
010220  000205                          RTS     R5              ;EXIT HERE ON STATUS ERROR
010222  017737  171272  001126  1$:     MOV     @DRVWCR,$BDDAT  ;GET WC
010230  001410                          BEQ     2$              ;BR IF ZERO
010232  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WCR ADRS
010240  005037  001124                  CLR     $GDDAT          ;EXPECTED 0
010244  062705  000006                  ADD     #6,R5           ;POINT TO THE WCR ER
010250  000205                          RTS     R5              ;EXIT HERE ON WCR ER
010252  011537  001124          2$:     MOV     (R5),$GDDAT     ;GET XFER #
010256  006337  001124                  ASL     $GDDAT          ;CONVERT TO WORD
010262  063737  001542  001124          ADD     DBUFP,$GDDAT    ;POINT TO LAST XFER +2
010270  017737  171226  001126          MOV     @DRVBAR,$BDDAT  ;GET BA
010276  042737  000001  001126          BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
010304  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
010312  001406                          BEQ     3$              ;BR IF SO
010314  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
010322  062705  000012                  ADD     #12,R5          ;POINT TO BAR ER
010326  000205                          RTS     R5              ;EXIT HERE ON BAR ER
010330  062705  000016          3$:     ADD     #16,R5          ;ALL OK - POINT TO GOOD EXIT
010334  000205                          RTS     R5              ;EXIT HERE IF NO ERRORS
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
                                ;FOR 199 LOCATIONS - THE LAST LOCATION IS LOADED WITH THE
                                ;#70707 WHICH SHOULD BE THE DATA WORD AVAILABLE IN THE DBR
                                ;AT THE COMPLETION OF A 200 WORD TRANSFER
                                ;;*****************************************************************
 7042  010336  012703  015316  LDBUF:  MOV     #DBUF,R3        ;GET BUFFER ADRS
```

```
7044  010342  012704  177776          1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
7045  010346  010423                  2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
7046  010350  005104                          COM     R4              ;MAKE INTO FLOATING 1
7047  010352  022703  016134                  CMP     #DBUF+616,R3    ;AT END OF BUFFER?
7048  010356  001003                          BNE     4$              ;BR IF NOT
7049  010360  012713  070707          3$:     MOV     #70707,(R3)     ;LOAD LAST DATVAR (SPECIAL)
7050  010364  000207                          RTS     PC              ;GET OUT
7051  010366  010423                  4$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 1)
7052  010370  022703  016134                  CMP     #DBUF+616,R3    ;AT END OF BUFFER?
7053  010374  001771                          BEQ     3$              ;BR IF SO
7054  010376  005104                          COM     R4              ;BACK TO FLOATING ZERO
7055  010400  006304                          ASL     R4              ;SHIFT LEFT
7056  010402  005204                          INC     R4              ;KEEP LSB SET
7057  010404  103356                          BCC     1$              ;GO RESET FLOATING PATRN
7058  010406  000757                          BR      2$              ;GO LOAD NEXT PATRN
7059
7060                                  ;;*****************************************************************
7061                                  ;THIS ROUTINE LOADS 'DBUF' WITH A UNIQUE FLOATING ZERO/ONE PATTERN
7062                                  ;(177776,0,1,0,177775,0,2,0,177773,0,4,0,177767,0,10,0 ETC.)
7063                                  ;IT IS USED WITH MAINT BIT SET (DATI/DATO SEQUENCE) - 200 LOCS
7064                                  ;ARE LOADED WITH THIS PATTERN
7065                                  ;;*****************************************************************
7066  010410  013703  001542          LDBUF1: MOV     DBUFP,R3        ;GET BUFFER ADRS
7067  010414  010305                          MOV     R3,R5           ;SAVE IN R5
7068  010416  062705  000620                  ADD     #620,R5         ;POINT TO END OF BUFFER
7069  010422  012704  177776          1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
7070  010426  010423                  2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
7071  010430  005023                          CLR     (R3)+           ;ZERO NEXT
7072  010432  005104                          COM     R4              ;SET UP FLOATING 1
7073  010434  010423                          MOV     R4,(R3)+        ;LOAD IT
7074  010436  005023                          CLR     (R3)+           ;ZERO NEXT
7075  010440  020503                          CMP     R5,R3           ;200 LOCS DONE?
7076  010442  001001                          BNE     3$              ;BR IF NOT
7077  010444  000207                          RTS     PC              ;GET OUT
7078  010446  005104                  3$:     COM     R4              ;BACK TO FLOATING ZERO
7079  010450  006304                          ASL     R4              ;SHIFT LEFT
7080  010452  005204                          INC     R4              ;KEEP LSB SET
7081  010454  103362                          BCC     1$              ;GO RESET FLOATING PATRN
7082  010456  000763                          BR      2$              ;GO FLOAT NEXT PATRN
7083
7084                                  ;;*****************************************************************
7085                                  ;THIS ROUTINE CHECKS 200 LOCATIONS IN "DBUF" FOR GOOD TRANSFERED
7086                                  ;DATA (#177377) ON 'DATO' TRANSFERS - IF AN ERROR IS DETECTED
7087                                  ;THE RETURN IS TO CALL +2 - IF NO ERROR THE RETURN IS TO CALL +4
7088                                  ;;*****************************************************************
7089  010460  012701  015316          CKDAT:  MOV     #DBUF,R1        ;GET BUFFER ADRS
7090  010464  022721  177377          1$:     CMP     #177377,(R1)+   ;DATA OK?
7091  010470  001410                          BEQ     2$              ;BR IF SO
7092  010472  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7093  010500  014137  001126                  MOV     -(R1),$BDDAT    ;GET ACTUAL DATA XFERED
7094  010504  010137  001120                  MOV     R1,$GDADR       ;GET MEMORY ADRS
7095  010510  000207                          RTS     PC              ;RETURN TO ERROR
7096  010512  022701  016136          2$:     CMP     #DBUF+620,R1    ;AT END OF 'DBUF'?
7097  010516  001362                          BNE     1$              ;BR IF MORE
7098  010520  062716  000002                  ADD     #2,(SP)         ;ADJUST STACK FOR GOOD RETURN
7099  010524  000207                          RTS     PC              ;GET OUT
```

```
7100
7101                                    ;;*****************************************************************
7102                                    ;THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
7103                                    ;DATA (177776,177776,1,1,177775,177775,2,2,177773,177773,ETC.)
7104                                    ;ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED
7105                                    ;BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 -
7106                                    ;IF NO ERROR THE RETURN IS TO CALL +10
7107                                    ;;*****************************************************************
7108  010526  012500                   CKDAT1: MOV     (R5)+,R0        ;GET # OF CHECKS
7109  010530  013702  001542                   MOV     DBUFP,R2        ;GET BUFFER ADRS
7110  010534  012701  177776           1$:     MOV     #177776,R1      ;SET UP FLOATING ZERO PATRN
7111  010540  005003                           CLR     R3              ;R3 SAYS WHEN TO SHIFT PATRN
7112  010542  020122                   2$:     CMP     R1,(R2)+        ;DATA OK?
7113  010544  001010                           BNE     3$              ;BR IF NOT
7114  010546  020122                           CMP     R1,(R2)+        ;DATA WRITTEN OK?
7115  010550  001006                           BNE     3$              ;BR IF NOT
7116  010552  162700  000002                   SUB     #2,R0           ;ACCOUNT FOR TWO ADRS'S
7117  010556  003015                           BGT     4$              ;BR IF MORE
7118  010560  062705  000004                   ADD     #4,R5           ;ADJUST FOR GOOD RETURN
7119  010564  000205                           RTS     R5              ;EXIT
7120  010566  014237  001126           3$:     MOV     -(R2),$BDDAT    ;GET BAD DATA
7121  010572  010237  001120                   MOV     R2,$GDADR       ;GET MEM ADRS
7122  010576  010137  001124                   MOV     R1,$GDDAT       ;LD EXPECTED DATA
7123  010602  013737  001526  001122         MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7124  010610  000205                           RTS     R5              ;RETURN TO ERROR
7125  010612  005101                   4$:     COM     R1              ;NOW EXPECT COMPLEMENT
7126  010614  005103                           COM     R3              ;TIME TO SHIFT?
7127  010616  001351                           BNE     2$              ;BR IF NOT
7128  010620  006301                           ASL     R1              ;SHIFT LEFT
7129  010622  005201                           INC     R1              ;KEEP LSB SET
7130  010624  103343                           BCC     1$              ;GO RESET FLOATING PATRN
7131  010626  000745                           BR      2$              ;DO NEXT
```

```
                                        .SBTTL  SYSMAC ROUTINES


                                .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
                                ;*

010630  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010634  100002                          BPL     1$              ;;BR IF YES
010636  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
010640  000430                          BR      3$              ;;LEAVE
010642  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
010644  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010650  122737  000001  001214          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
010656  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
010660  132737  000100  001215          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
010666  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
010670  010037  010700                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
010674  004737  011120                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
010700  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
010702  132737  000040  001215  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
010710  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
010712  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010714  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010716  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010720  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
010722  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
010726  000002                          RTI                     ;;RETURN
010730  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
010734  001430                          BEQ     8$
010736  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
010742  001006                          BNE     5$
010744  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
010746  104401                          TYPE                    ;;TYPE A CR AND LF
010750  001171                          $CRLF
010752  105037  011106                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
010756  000755                          BR      2$              ;;GET NEXT CHARACTER
010760  004737  011042          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
010764  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
010770  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
010772  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
010776  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
011002  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1)  011004  004737  011042                JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  011010  105337  011106                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  011014  000770                        BR      7$              ;;LOOP
(1)
(1)                                 ;HORIZONTAL TAB PROCESSOR
(1)
(1)  011016  112716  000040        8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  011022  004737  011042        9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  011026  132737  000007  011106        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  011034  001372                        BNE     9$              ;;TAB STOP
(1)  011036  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
(1)  011040  000724                        BR      2$              ;;GET NEXT CHARACTER
(1)  011042  105777  170102        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  011046  100375                        BPL     $TYPEC
(1)  011050  116677  000002  170074        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)
(1)  011056  122766  000015  000002  CMPB  #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
(1)  011064  001003                        BNE     1$              ;;BRANCH IF NO
(1)  011066  105037  011106                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  011072  000406                        BR      $TYPEX          ;;EXIT
(1)  011074  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  011102  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
(1)  011104  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  011106  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  011110  000207                $TYPEX: RTS     PC
(1)
(1)
(1)
7136                                .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)  011112  112737  000001  011356 $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  011120  112737  000001  011354 $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  011126  000403                        BR      $ATYC
(1)  011130  112737  000001  011356 $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  011136                        $ATYC:
(3)  011136  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  011140  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  011142  105737  011354                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  011146  001450                        BEQ     5$              ;;IF NOT:  BR
(1)  011150  122737  000001  001214        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  011156  001031                        BNE     3$              ;;IF NOT:  BR
(1)  011160  132737  000100  001215        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  011166  001425                        BEQ     3$              ;;IF NOT:  BR
(1)  011170  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  011174  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  011202  005737  001174        1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  011206  001375                        BNE     1$              ;;IF NOT:  WAIT
(1)  011210  010037  001210                MOV     R0,$MSGAD
(1)                                 ;;PUT ADDR IN MAILBOX
(1)  011214  105720                2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  011216  001376                        BNE     2$
(1)  011220  163700  001210                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  011224  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  011226  010037  001212                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  011232  012737  000004  001174        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
```

```
(1)  011240  000413                          BR      5$
(1)  011242  017637  000004  011266  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  011250  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  011256  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  011262  004737  010630                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  011266  000000                  4$:     .WORD   0
(1)  011270                          5$:
(1)  011270  105737  011356          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  011274  001416                          BEQ     12$             ;;IF NOT:  BR
(1)  011276  005737  001214                  TST     $ENV            ;;RUNNING UNDER APT?
(1)  011302  001413                          BEQ     12$             ;;IF NOT:  BR
(1)  011304  005737  001174          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  011310  001375                          BNE     11$             ;;IF NOT:  WAIT
(1)  011312  017637  000004  001176          MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  011320  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  011326  005237  001174                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  011332  105037  011356          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  011336  105037  011355                  CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  011342  105037  011354                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  011346  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  011350  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  011352  000207                          RTS     PC              ;;RETURN
(1)  011354     000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  011355     000                  $LFLG:  .BYTE   0
(1)                                          ;;LOG FLAG
(1)  011356     000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          011360                          .EVEN
(1)          000200                  APTSIZE=200
(1)          000001                  APTENV=001
(1)          000100                  APTSPOOL=100
(1)          000040                  APTCSUP=040
7137                                 .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                                  ;;**************************************************************************
(1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                                  ;*CALL:
(1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                                  ;*      .BYTE   M               ;;M=1 OR 0
(1)                                  ;*                                      ;;1=TYPE LEADING ZEROS
(1)                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                                  ;*
(1)                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                  ;*$TYPOS OR $TYPOC
(1)                                  ;*CALL:
(1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                  ;*
(1)                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                  ;*CALL:
(1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
```

```
(1)  011360  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  011364  116637  000001  011603          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  011372  112637  011605                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  011376  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  011402  000406                          BR      $TYPON
(1)  011404  112737  000001  011603  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  011412  112737  000006  011605          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  011420  112737  000005  011602  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  011426  010346                          MOV     R3,-(SP)        ;;SAVE R3
(1)  011430  010446                          MOV     R4,-(SP)        ;;SAVE R4
(1)  011432  010546                          MOV     R5,-(SP)        ;;SAVE R5
(1)  011434  113704  011605                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  011440  005404                          NEG     R4
(1)  011442  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  011446  110437  011604                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  011452  113704  011603                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  011456  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  011462  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  011464  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  011466  000404                          BR      3$              ;;GO DO MSB
(1)  011470  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  011472  006105                          ROL     R5
(1)  011474  006105                          ROL     R5
(1)  011476  010503                          MOV     R5,R3
(1)  011500  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  011502  105337  011604                  DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  011506  100016                          BPL     7$              ;;BR IF NO
(1)  011510  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
(1)  011514  001002                          BNE     4$              ;;TEST FOR 0
(1)  011516  005704                          TST     R4              ;;SUPPRESS THIS 0?
(1)  011520  001403                          BEQ     5$              ;;BR IF YES
(1)  011522  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  011524  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  011530  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  011534  110337  011600                  MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  011540  104401  011600                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  011544  105337  011602          7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  011550  003347                          BGT     2$              ;;BR IF MORE TO DO
(1)  011552  002402                          BLT     6$              ;;BR IF DONE
(1)  011554  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  011556  000744                          BR      2$              ;;GO DO THE LAST DIGIT
(1)  011560  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  011562  012604                          MOV     (SP)+,R4        ;;RESTORE R4
(1)  011564  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  011566  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  011574  012616                          MOV     (SP)+,(SP)
(1)  011576  000002                          RTI                     ;;RETURN
(1)  011600     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  011601     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  011602     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  011603     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  011604  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7138                                 .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
```

```
(1)                                     ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                     ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                     ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                     ;*REPLACED WITH SPACES.
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                     ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(1)  011606                             $TYPDS:
(3)  011606  010046                             MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  011610  010146                             MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  011612  010246                             MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  011614  010346                             MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  011616  010546                             MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  011620  012746  020200                      MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  011624  016605  000020                      MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  011630  100004                              BPL     1$              ;;BR IF INPUT IS POS.
(1)  011632  005405                              NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  011634  112766  000055  000001              MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  011642  005000                      1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  011644  012703  012022                      MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  011650  112723  000040                      MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  011654  005002                      2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  011656  016001  012012                      MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  011662  160105                      3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  011664  002402                              BLT     4$              ;;BR IF DONE
(1)  011666  005202                              INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  011670  000774                              BR      3$
(1)  011672  060105                      4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  011674  005702                              TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  011676  001002                              BNE     5$              ;;FALL THROUGH IF 0
(1)  011700  105716                              TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  011702  100407                              BMI     7$              ;;BR IF YES
(1)  011704  106316                      5$:     ASLB    (SP)            ;;MSD?
(1)  011706  103003                              BCC     6$              ;;BR IF NO
(1)  011710  116663  000001  177777              MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  011716  052702  000060              6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  011722  052702  000040              7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  011726  110223                              MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  011730  005720                              TST     (R0)+           ;;JUST INCREMENTING
(1)  011732  020027  000010                      CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  011736  002746                              BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  011740  003002                              BGT     8$              ;;GO TO EXIT
(1)  011742  010502                              MOV     R5,R2           ;;GET THE LSD
(1)  011744  000764                              BR      6$              ;;GO CHANGE TO ASCII
(1)  011746  105726                      8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  011750  100003                              BPL     9$              ;;BR IF NO
(1)  011752  116663  177777  177776              MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  011760  105013                      9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  011762  012605                              MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  011764  012603                              MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  011766  012602                              MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  011770  012601                              MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  011772  012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  011774  104401  012022                      TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  012000  016666  000002  000004              MOV     2(SP),4(SP)     ;;ADJUST THE STACK
```

```
                                                                I 4
  (1)  012006  012616                   MOV     (SP)+,(SP)
  (1)  012010  000002                   RTI                             ;;RETURN TO USER
  (1)  012012  023420           $DTBL:  10000.
  (1)  012014  001750                   1000.
  (1)  012016  000144                   100.
  (1)  012020  000012                   10.
  (1)  012022  000004           $DBLK:  .BLKW   4
7139
  (1)                           .SBTTL  ERROR HANDLER ROUTINE
  (1)
  (2)                           ;;**********************************************************************
  (1)                           ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
  (1)                           ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
  (1)                           ;*AND GO TO SWRCK ON ERROR
  (1)                           ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
  (1)                           ;*SW15=1        HALT ON ERROR
  (1)                           ;*SW13=1        INHIBIT ERROR TYPEOUTS
  (1)                           ;*SW10=1        BELL ON ERROR
  (1)                           ;*SW09=1        LOOP ON ERROR
  (1)                           ;*CALL
  (1)                           ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  (1)
  (1)  012032                   $ERROR:
  (1)  012032  104407                   CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
  (2)  012034  104407                   CKSWR           ;GO LOOK FOR SWR CHANGE
  (1)  012036  105237  001103   7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
  (1)  012042  001775                   BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
  (1)  012044  013777  001102  167070   MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
  (1)  012052  032777  002000  167060   BIT     #BIT10,@SWR             ;;BELL ON ERROR?
  (1)  012060  001402                   BEQ     1$                      ;;NO - SKIP
  (1)  012062  104401  001164           TYPE    ,$BELL                  ;;RING BELL
  (1)  012066  005237  001112   1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
  (1)  012072  011637  001116           MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
  (1)  012076  162737  000002  001116   SUB     #2,$ERRPC
  (1)  012104  117737  167006  001114   MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
  (1)  012112  032777  020000  167020   BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
  (1)  012120  001004                   BNE     20$                     ;;SKIP TYPEOUTS
  (1)  012122  004737  012222           JSR     PC,SWRCK                ;;GO TO USER ERROR ROUTINE
  (1)  012126  104401  001171           TYPE    ,$CRLF
  (1)  012132                   20$:
  (1)  012132  122737  000001  001214   CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
  (1)  012140  001007                   BNE     2$                      ;;NO,SKIP APT ERROR REPORT
  (1)  012142  113737  001114  012154   MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
  (1)  012150  004737  011130           JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
  (1)  012154     000           21$:    .BYTE   0
  (1)  012155     000                   .BYTE   0
  (1)  012156  000777           22$:    BR      22$                     ;;APT ERROR LOOP
  (1)  012160  005777  166754   2$:     TST     @SWR                    ;;HALT ON ERROR
  (1)  012164  100002                   BPL     3$                      ;;SKIP IF CONTINUE
  (1)  012166  000000                   HALT                            ;;HALT ON ERROR!
  (1)  012170  104407                   CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
  (1)  012172  032777  001000  166740   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
  (1)  012200  001402                   BEQ     4$                      ;;BR IF NO
  (1)  012202  013716  001110           MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
  (1)  012206  005737  001162   4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
  (1)  012212  001402                   BEQ     5$                      ;;BR IF NONE
  (1)  012214  013716  001162           MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
                                        5$:
  (1)  012220                                   RTI                             ;;RETURN
  (1)  012220  000002
 7140                                   ;;*******************************************************************
 7141                                   ;GO TYPE ERROR
 7142                                   ;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
 7143                                   ;;*******************************************************************
 7144  012222  113737  001102  001534  SWRCK:  MOVB    $TSTNM,TSTNUM   ;SET UP TEST # ON ER
 7145  012230  004737  012240                 JSR     PC,$ERRTYP      ;GO TYPE ERROR
 7146  012234  104407                         CKSWR                   ;GO LOOK FOR SWR CHANGE
 7147  012236  000207                         RTS     PC              ;RETURN TO ERROR HANDLER
 7148                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
  (1)
  (2)                                   ;;*******************************************************************
  (1)                                   ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
  (1)                                   ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
  (1)                                   ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
  (1)
  (1)  012240                           $ERRTYP:
  (1)  012240  104401  001171                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  012244  010046                           MOV     R0,-(SP)        ;;SAVE R0
  (1)  012246  005000                           CLR     R0              ;;PICKUP THE ITEM INDEX
  (1)  012250  153700  001114                   BISB    @#$ITEMB,R0
  (1)  012254  001004                           BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
  (1)                                                                   ;;TYPE THE PC OF THE ERROR
  (2)  012256  013746  001116                   MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
  (2)                                                                   ;;ERROR ADDRESS
  (2)  012262  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  (1)  012264  000426                           BR      6$              ;;GET OUT
  (1)  012266  005300                   1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
  (1)  012270  006300                           ASL     R0              ;;      WORK FOR THE ERROR TABLE
  (1)  012272  006300                           ASL     R0
  (1)  012274  006300                           ASL     R0
  (1)  012276  062700  001320                   ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
  (1)  012302  012037  012312                   MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
  (1)  012306  001404                           BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
  (1)  012310  104401                           TYPE                    ;;TYPE THE "ERROR MESSAGE"
  (1)  012312  000000                   2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
  (1)  012314  104401  001171                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  012320  012037  012330           3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
  (1)  012324  001404                           BEQ     5$              ;;SKIP TYPEOUT IF 0
  (1)  012326  104401                           TYPE                    ;;TYPE THE "DATA HEADER"
  (1)  012330  000000                   4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
  (1)  012332  104401  001171                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  012336  011000                   5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
  (1)  012340  001004                           BNE     7$              ;;GO TYPE THE DATA
  (1)  012342  012600                   6$:     MOV     (SP)+,R0        ;;RESTORE R0
  (1)
  (1)  012344  104401  001171                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  012350  000207                           RTS     PC              ;;RETURN
  (1)  012352                           7$:
  (2)  012352  013046                           MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
  (2)  012354  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  (1)  012356  005710                           TST     (R0)            ;;IS THERE ANOTHER NUMBER?
  (1)  012360  001770                           BEQ     6$              ;;BR IF NO
  (1)  012362  104401  012370                   TYPE    ,8$             ;;TYPE TWO(2) SPACES
  (1)  012366  000771                           BR      7$              ;;LOOP
```

```
   (1)  012370  020040      000          8$:     .ASCIZ  / /             ;;TWO(2) SPACES
   (1)           012374                          .EVEN
  7149                                  .SBTTL  SCOPE HANDLER ROUTINESTARS
   (1)                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
   (1)                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
   (1)                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
   (1)                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
   (1)                                  ;*SW14=1        LOOP ON TEST
   (1)                                  ;*SW11=1        INHIBIT ITERATIONS
   (1)                                  ;*SW09=1        LOOP ON ERROR
   (1)                                  ;*SW08=1        LOOP ON TEST IN SWR<5:0>
   (1)                                  ;*CALL
   (1)                                  ;*      SCOPE           ;;SCOPE=IOT
   (1)
   (1)  012374                          $SCOPE:
   (1)  012374  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
   (2)  012376  104407                          CKSWR                   ;GO LOOK FOR SWR CHANGE
   (1)  012400  032777  040000  166532  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
   (1)  012406  001117                          BNE     $OVER           ;;YES IF SW14=1
   (1)                                  ;#####START OF CODE FOR THE XOR TESTER#####
   (1)  012410  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
   (1)                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
   (1)  012412  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
   (1)  012416  012737  012436  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
   (1)  012424  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
   (1)  012430  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
   (1)  012434  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
   (1)  012436  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
   (1)  012440  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
   (1)  012444  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
   (1)  012446                          6$:;#####END OF CODE FOR THE XOR TESTER#####
   (1)  012446  032777  000400  166464          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
   (1)  012454  001407                          BEQ     2$              ;;BR IF NO
   (1)  012456  017746  166456                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
   (1)  012462  042716  000300                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
   (1)  012466  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
   (1)  012472  001465                          BEQ     $OVER           ;;BR IF YES
   (1)  012474  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
   (1)  012500  001421                          BEQ     3$              ;;BR IF NO
   (1)  012502  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
   (1)  012510  101015                          BHI     3$              ;;BR IF NO
   (1)  012512  032777  001000  166420          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
   (1)  012520  001404                          BEQ     4$              ;;BR IF NO
   (1)  012522  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
   (1)  012530  000446                          BR      $OVER
   (1)  012532  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
   (1)  012536  005037  001160                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
   (1)  012542  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
   (1)  012544  032777  004000  166366  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
   (1)  012552  001011                          BNE     1$              ;;BR IF YES
   (1)  012554  005737  001202                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
   (1)  012560  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
   (1)  012562  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
   (1)  012566  023737  001160  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
   (1)  012574  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
   (1)  012576  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
```

```
                                                                    L 4
   (1)  012604  013737  012662  001160          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
   (1)  012612  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
   (1)  012616  113737  001102  001200          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
   (1)  012624  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
   (1)  012630  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
   (1)  012634  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
   (1)  012640  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
   (1)  012646  013777  001102  166266  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
   (1)  012654  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
   (1)  012660  000002                          RTI                     ;;FIXES PS
   (1)  012662  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
  7150                                  .SBTTL  TTY INPUT ROUTINE
   (1)
   (2)                                  ;;**************************************************************************
   (1)
   (1)
   (1)                                  .ENABL  LSB
   (1)
   (2)                                  ;;**************************************************************************
   (1)                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
   (1)                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
   (1)                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
   (1)                                  ;*WHEN OPERATING IN TTY FLAG MODE.
   (1)  012664  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
   (1)  012672  001074                          BNE     15$             ;;BRANCH IF NO
   (1)  012674  105777  166244                  TSTB    @$TKS           ;;CHAR THERE?
   (1)  012700  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
   (1)  012702  117746  166240                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
   (1)  012706  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
   (1)  012712  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
   (1)  012716  001062                          BNE     15$             ;;NO, RETURN TO USER
   (1)  012720  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
   (1)  012726  001456                          BEQ     15$             ;;BRANCH IF YES
   (1)
   (1)  012730  104401  013411                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
   (1)  012734  104401  013416          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
   (2)  012740  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
   (2)  012744  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
   (1)  012746  104401  013427                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
   (1)  012752  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
   (1)  012754  005046                          CLR     -(SP)
   (1)                                          ;;THE NEW SWR
   (1)  012756  105777  166162          7$:     TSTB    @$TKS           ;;CHAR THERE?
   (1)  012762  100375                          BPL     7$              ;;IF NOT TRY AGAIN
   (1)
   (1)  012764  117746  166156                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
   (1)  012770  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
   (1)
   (1)
   (1)
   (1)  012774  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
   (1)  013000  001005                          BNE     10$             ;;BRANCH IF NOT
   (1)  013002  104401  013404                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
   (1)  013006  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
   (1)  013012  000757                          BR      19$             ;;LET'S TRY IT AGAIN
   (1)
```

```
(1)
(1)  013014  021627  000015          10$:   CMP     (SP),#15        ;;IS IT A <CR>?
(1)  013020  001022                         BNE     16$             ;;BRANCH IF NO
(1)  013022  005766  000004                 TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  013026  001403                         BEQ     11$             ;;BRANCH IF YES
(1)  013030  016677  000002  166102         MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  013036  062706  000006          11$:   ADD     #6,SP           ;;CLEAR UP STACK
(1)  013042  104401  001171          14$:   TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  013046  123727  001135  000001         CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  013054  001003                         BNE     15$             ;;BRANCH IF NOT
(1)  013056  012777  000100  166060         MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  013064  000002                  15$:   RTI                     ;;RETURN
(1)  013066  004737  011042          16$:   JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  013072  021627  000060                 CMP     (SP),#60        ;;CHAR < 0?
(1)  013076  002420                         BLT     18$             ;;BRANCH IF YES
(1)  013100  021627  000067                 CMP     (SP),#67        ;;CHAR > 7?
(1)  013104  003015                         BGT     18$             ;;BRANCH IF YES
(1)  013106  042726  000060                 BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  013112  005766  000002                 TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  013116  001403                         BEQ     17$             ;;BRANCH IF YES
(1)  013120  006316                         ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  013122  006316                         ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  013124  006316                         ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  013126  005266  000002          17$:   INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  013132  056616  177776                 BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  013136  000707                         BR      7$              ;;GET THE NEXT ONE
(1)  013140  104401  001170          18$:   TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  013144  000720                         BR      20$             ;;SIMULATE CONTROL-U
(1)                                  .DSABL  LSB
(1)
(1)
(2)                                  ;;**************************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                  ;
(1)
(1)  013146  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  013150  016666  000004  000002         MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  013156  105777  165762          1$:    TSTB    @$TKS           ;;WAIT FOR
(1)  013162  100375                         BPL     1$              ;;A CHARACTER
(1)  013164  117766  165756  000004         MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  013172  042766  177600  000004         BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  013200  026627  000004  000023         CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  013206  001013                         BNE     3$              ;;BRANCH IF NO
(1)  013210  105777  165730          2$:    TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  013214  100375                         BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  013216  117746  165724                 MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  013222  042716  177600                 BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  013226  022627  000021                 CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  013232  001366                         BNE     2$              ;;IF NOT DISCARD IT
(1)  013234  000750                         BR      1$              ;;YES, RESUME
(1)  013236  026627  000004  000140  3$:    CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  013244  002407                         BLT     4$              ;;BRANCH IF YES
```

```
(1)  013246  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  013254  003003                          BGT     4$              ;;BRANCH IF YES
(1)  013256  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  013264  000002                  4$:     RTI                     ;;GO BACK TO USER
(2)                                        ;;******************************************************************
(1)                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                        ;*CALL:
(1)                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  013266  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  013270  012703  013374          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  013274  022703  013404          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
(1)  013300  101405                          BLOS    4$              ;;BR IF YES
(1)  013302  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  013304  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  013306  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  013312  001003                          BNE     3$              ;;SKIP IF NOT
(1)  013314  104401  001170          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  013320  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  013322  111337  013372          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  013326  104401  013372                  TYPE    ,9$
(1)  013332  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  013336  001356                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  013340  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  013344  104401  001172                  TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  013350  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  013352  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  013354  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  013362  012766  013374  000004          MOV     #$TTYIN,4(SP)
(1)  013370  000002                          RTI                     ;;RETURN
(1)  013372     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  013373     000                          .BYTE   0               ;;TERMINATOR
(1)  013374  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  013404  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
(1)  013411     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
(1)  013416  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  013424  020075     000
(1)  013427     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
(1)  013434  036440  000040
7151                                       .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(2)                                        ;;******************************************************************
(1)                                        ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                        ;*CHANGE IT TO BINARY.
(1)                                        ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
(1)                                        ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
(1)                                        ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
(1)                                        ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
(1)                                        ;*CALL:
(1)                                        ;*      RDOCT                   ;;READ AN OCTAL NUMBER
(1)                                        ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                        ;*BITS ARE IN $HIOCT                            ;;HIGH ORDER
(1)
(1)  013440  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  013442  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
```

```
    (3)  013450  010046                1$:     MOV     R0,-(SP)        ;;PUSH R0 ON STACK
    (3)  013452  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    (3)  013454  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
    (1)  013456  104411                        RDLIN                   ;;READ AN ASCIZ LINE
    (1)  013460  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
    (1)  013462  010037  013566                MOV     R0,5$           ;;AND SAVE IT
    (1)  013466  005001                        CLR     R1              ;;CLEAR DATA WORD
    (1)  013470  005002                        CLR     R2
    (1)  013472  112046                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
    (1)  013474  001420                        BEQ     3$              ;;IF ZERO GET OUT
    (1)  013476  122716  000060                CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
    (1)  013502  003026                        BGT     4$              ;;IS AN OCTAL DIGIT
    (1)  013504  122716  000067                CMPB    #'7,(SP)
    (1)  013510  002423                        BLT     4$
    (1)  013512  006301                        ASL     R1              ;;*2
    (1)  013514  006102                        ROL     R2
    (1)  013516  006301                        ASL     R1              ;;*4
    (1)  013520  006102                        ROL     R2
    (1)  013522  006301                        ASL     R1              ;;*8
    (1)  013524  006102                        ROL     R2
    (1)  013526  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
    (1)  013532  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
    (1)  013534  000756                        BR      2$              ;;LOOP
    (1)  013536  005726                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
    (1)  013540  010166  000012                MOV     R1,12(SP)       ;;SAVE THE RESULT
    (1)  013544  010237  013576                MOV     R2,$HIOCT
    (3)  013550  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
    (3)  013552  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
    (3)  013554  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
    (1)  013556  000002                        RTI                     ;;RETURN
    (1)  013560  005726                4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
    (1)  013562  105010                        CLRB    (R0)            ;;SET A TERMINATOR
    (1)  013564  104401                        TYPE                    ;;TYPE UP THRU THE BAD CHAR.
    (1)  013566  000000                5$:     .WORD   0
    (1)  013570  104401  001170                TYPE    ,$QUES          ;;"?" "CR" & "LF"
    (1)  013574  000730                        BR      1$              ;;TRY AGAIN
    (1)  013576  000000                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
    (1)  013600                        S:
7152                                   .SBTTL  POWER DOWN AND UP ROUTINES
    (1)
    (2)                                ;;**************************************************************************
    (1)                                ;POWER DOWN ROUTINE
    (1)  013600  012737  013744  000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
    (1)  013606  012737  000300  000026        MOV    #PR6,@#PWRVEC+2 ;;PRIO:6
    (3)  013614  010046                        MOV    R0,-(SP)        ;;PUSH R0 ON STACK
    (3)  013616  010146                        MOV    R1,-(SP)        ;;PUSH R1 ON STACK
    (3)  013620  010246                        MOV    R2,-(SP)        ;;PUSH R2 ON STACK
    (3)  013622  010346                        MOV    R3,-(SP)        ;;PUSH R3 ON STACK
    (3)  013624  010446                        MOV    R4,-(SP)        ;;PUSH R4 ON STACK
    (3)  013626  010546                        MOV    R5,-(SP)        ;;PUSH R5 ON STACK
    (3)  013630  017746  165304                MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
    (1)  013634  010637  013750                MOV    SP,$SAVR6       ;;SAVE SP
    (1)  013640  012737  013652  000024        MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
    (1)  013646  000000                        HALT
    (1)  013650  000776                        BR     .-2             ;;HANG UP
    (1)
```

```
(2)                                     ;;**********************************************************************
(1)                                     ;POWER UP ROUTINE
(1)  013652  012737  013744  000024     $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1)  013660  013706  013750             MOV     $SAVR6,SP        ;;GET SP
(1)  013664  005037  013750             CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
(1)  013670  005237  013750             1$:     INC     $SAVR6           ;;WAIT FOR THE INC
(1)  013674  001375                     BNE     1$               ;;OF  WORD
(3)  013676  012677  165236             MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
(3)  013702  012605                     MOV     (SP)+,R5         ;;POP STACK INTO R5
(3)  013704  012604                     MOV     (SP)+,R4         ;;POP STACK INTO R4
(3)  013706  012603                     MOV     (SP)+,R3         ;;POP STACK INTO R3
(3)  013710  012602                     MOV     (SP)+,R2         ;;POP STACK INTO R2
(3)  013712  012601                     MOV     (SP)+,R1         ;;POP STACK INTO R1
(3)  013714  012600                     MOV     (SP)+,R0         ;;POP STACK INTO R0
(1)  013716  012737  013600  000024     MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1)  013724  012737  000300  000026     MOV     #PR6,@#PWRVEC+2  ;;PRIO:6
(1)  013732  104401                     TYPE                     ;;REPORT THE POWER FAILURE
(1)  013734  013752                     $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
(1)  013736  012716                     MOV     (PC)+,(SP)       ;;RESTART AT RESTRT
(1)  013740  002406                     $PWRAD: .WORD   RESTRT           ;;RESTART ADDRESS
(1)  013742  000002                     RTI
(1)  013744  000000                     $ILLUP: HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
(1)  013746  000776                     BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
(1)  013750  000000                     $SAVR6: 0                        ;;PUT THE SP HERE
7153 013752  005015  042522  052123     PWRMSG: .ASCIZ  <15><12>/RESTARTED FROM PWR FAIL/
     013760  051101  042524  020104
     013766  051106  046517  050040
     013774  051127  043040  044501
     014002  000114
7154                                    .EVEN
7155                                    .SBTTL  TRAP DECODER
 (1)                                    
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
 (1)                                    ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
 (1)                                    ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
 (1)                                    ;*GO TO THAT ROUTINE.
 (1)                                    
 (1)  014004  010046                     $TRAP:  MOV     R0,-(SP)         ;;SAVE R0
 (1)  014006  016600  000002             MOV     2(SP),R0         ;;GET TRAP ADDRESS
 (1)  014012  005740                     TST     -(R0)            ;;BACKUP BY 2
 (1)  014014  111000                     MOVB    (R0),R0          ;;GET RIGHT BYTE OF TRAP
 (1)  014016  006300                     ASL     R0               ;;POSITION FOR INDEXING
 (1)  014020  016000  014040             MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
 (1)  014024  000200                     RTS     R0               ;;GO TO ROUTINE
 (1)                                    
 (1)                                    
 (1)                                    ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
 (1)                                    
 (1)  014026  011646                     $TRAP2: MOV     (SP),-(SP)       ;;MOVE THE PC DOWN
 (1)  014030  016666  000004  000002     MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
 (1)  014036  000002                     RTI                      ;;RESTORE THE PSW
 (1)                                    
 (3)                                    .SBTTL  TRAP TABLE
 (3)                                    
 (3)                                    ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
```

```
   (3)                          ;*BY THE "TRAP" INSTRUCTION.
   (3)
   (3)                          ;       ROUTINE
   (3)                          ;       -------
   (3)  014040  014026          $TRPAD: .WORD   $TRAP2
   (3)  014042  010630                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
   (3)  014044  011404                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
   (3)  014046  011360                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
   (3)  014050  011420                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
   (3)  014052  011606                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
   (1)
   (3)  014054  012734                  $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
   (1)
   (3)  014056  012664                  $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
   (3)  014060  013146                  $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
   (3)  014062  013266                  $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
   (3)  014064  013440                  $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
 7156
 7157                           .SBTTL  ASCII MESSAGES
 7158  014066  042522  020107  044524  EM1:    .ASCIZ  /REG TIMEOUT ER/
       014074  042515  052517  020124
       014102  051105     000
 7159  014105     122  043505  051040  EM2:    .ASCIZ  'REG READ/WRITE ER'
       014112  040505  027504  051127
       014120  052111  020105  051105
       014126     000
 7160  014127     102  051525  051040  EM3:    .ASCIZ  /BUS RESET ER/
       014134  051505  052105  042440
       014142  000122
 7161  014144  047106  052103  041040  EM4:    .ASCIZ  /FNCT BITS FAILED TO SET STAT BITS/
       014152  052111  020123  040506
       014160  046111  042105  052040
       014166  020117  042523  020124
       014174  052123  052101  041040
       014202  052111  000123
 7162  014206  042522  042101  020131  EM5:    .ASCIZ  /READY INTR FAILURE/
       014214  047111  051124  043040
       014222  044501  052514  042522
       014230     000
 7163  014231     122  040505  054504  EM6:    .ASCIZ  /READY CLR OR SET ER/
       014236  041440  051114  047440
       014244  020122  042523  020124
       014252  051105     000
 7164  014255     123  040524  052524  EM7:    .ASCIZ  /STATUS ER ON XFER/
       014262  020123  051105  047440
       014270  020116  043130  051105
       014276     000
 7165  014277     127  051117  020104  EM10:   .ASCIZ  /WORD COUNT ER ON XFER/
       014304  047503  047125  020124
       014312  051105  047440  020116
       014320  043130  051105     000
 7166  014325     102  043125  042506  EM11:   .ASCIZ  /BUFFER ADRS ER ON XFER/
       014332  020122  042101  051522
       014340  042440  020122  047117
       014346  054040  042506  000122
 7167  014354  040504  040524  042440  EM12:   .ASCIZ  /DATA ER FROM MEM/
```

```
                014362  020122  051106  046517
                014370  046440  046505     000
 7168   014375     104  052101  020101  EM13:   .ASCIZ  /DATA ER TO MEM/
                014402  051105  052040  020117
                014410  042515  000115
 7169   014414  044523  043516  042514  EM14:   .ASCIZ  /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
                014422  041440  041531  042514
                014430  047440  043106  042040
                014436  042111  047040  052117
                014444  046040  041517  020113
                014452  052517  020124  050103
                014460  000125
 7170   014462  044523  043516  042514  EM15:   .ASCIZ  /SINGLE CYCLE ON LOCKED OUT CPU/
                014470  041440  041531  042514
                014476  047440  020116  047514
                014504  045503  042105  047440
                014512  052125  041440  052520
                014520     000
 7171   014521     015  050012  042514  WARN:   .ASCII  <15><12>/PLEASE DISABLE "REV11" MEMORY REFRESH OPTION/
                014526  051501  020105  044504
                014534  040523  046102  020105
                014542  051042  053105  030461
                014550  020042  042515  047515
                014556  054522  051040  043105
                014564  042522  044123  047440
                014572  052120  047511     116
 7172   014577     015  040412  042116          .ASCIZ  <15><12>/AND ENABLE PROCESSOR MEMORY REFRESH   /
                014604  042440  040516  046102
                014612  020105  051120  041517
                014620  051505  047523  020122
                014626  042515  047515  054522
                014634  051040  043105  042522
                014642  044123  020040  000040
 7173   014650  042516  020130  047514  EM16:   .ASCIZ  /NEX LOGIC ER/
                014656  044507  020103  051105
                014664     000
 7174   014665     103  041531  042514  EM17:   .ASCIZ  /CYCLE FAILED TO CLK DBR (IN)/
                014672  043040  044501  042514
                014700  020104  047524  041440
                014706  045514  042040  051102
                014714  024040  047111  000051
 7175   014722  040504  040524  042440  EM20:   .ASCIZ  "DATA ER FROM I/O PAGE (XCSR)"
                014730  020122  051106  046517
                014736  044440  047457  050040
                014744  043501  020105  054050
                014752  051503  024522     000
 7176   014757     105  051122  041520  DH1:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  EXPCT   RCVD/
                014764  020040  052040  052123
                014772  052516  020115  041040
                015000  051525  042101  020122
                015006  042440  050130  052103
                015014  020040  051040  053103
                015022  000104
 7177   015024  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  ADRS    EXPCT   RCVD/
                015032  020040  051524  047124
                015040  046525  020040  052502
```

```
                015046  040523  051104  020040
                015054  042101  051522  020040
                015062  020040  054105  041520
                015070  020124  020040  041522
                015076  042126     000
 7178           015101     105  051122  041520  DH3:    .ASCIZ  /ERRPC   TSTNUM  BUSADR/
                015106  020040  052040  052123
                015114  052516  020115  041040
                015122  051525  042101  000122
 7179
 7180                                            .EVEN
 7181           015130  001116  001534  001122  DT1:    $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
                015136  001124  001126  000000
 7182           015144  001116  001534  001122  DT2:    $ERRPC,TSTNUM,$BDADR,$GDADR,$GDDAT,$BDDAT,0
                015152  001120  001124  001126
                015160  000000
 7183           015162  001116  001534  001122  DT3:    $ERRPC,TSTNUM,$BDADR,0
                015170  000000
 7184           015172  005015  051511  052040  KDF11B: .ASCIZ  <15><12>/IS THE PROCESSOR KDF11-B?  /
                015200  042510  050040  047522
                015206  042503  051523  051117
                015214  045440  043104  030461
                015222  041055  020077  000040
 7185           015230  005015  051124  047101  IOTEST: .ASCIZ  <15><12>.TRANSFERS FOR I/O PAGE?  .
                015236  043123  051105  020123
                015244  047506  020122  027511
                015252  020117  040520  042507
                015260  020077  000040
 7186           015264  005015  042101  051104  IOADR:  .ASCIZ  <15><12>\ADDRESS OF I/O PAGE?  \
                015272  051505  020123  043117
                015300  044440  047457  050040
                015306  043501  037505  020040
                015314     000
 7187                   015316                           .EVEN
 7188                                           ;;*****************************************************************
 7189                                           ;'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
 7190                                           ;NPR OPERATIONS - IT IS 200 WORDS LONG
 7191                                           ;;*****************************************************************
 7192           015316  000000                  DBUF:   0               ;1ST ADRS OF DATA BUFFER
```

```
7194                                    ;;THE FOLLOWING MACRO CALL TO CNMAC2.SML WAS ADDED TO MAKE
7195                                    ;;THIS DIAGNOSTIC SPECIFIC TO 11/21 PROCESSOR AND WAS RENAMED
7196                                    ;;FROM CVDRABO TO CNDRAAO. ALSO ASSEMBLED WITH CNMAC2.SML.
7197          015320                    POINT=.                 ;SAVE POINTER
 (1)          000100                    .=100
 (1)  000100  015320                    $CLKVEC                 ;LKVEC  HANDLER
 (1)  000102  000300                    300                     ;INTERRUPT HANDLER PRI
 (1)          000140                    .=140                   ;BRKVEC
 (1)  000140  170000                    170000                  ;ODT START ADDRESS
 (1)  000142  000300                    300                     ;PRIORITY
 (1)          015320                    .=POINT                 ;RESTORE POINTER
 (1)  015320  104401  015326            $CLKVEC:        TYPE,CLKMES
 (1)  015324  000000                                    HALT
 (1)  015326  005015  045514  042526    CLKMES:  .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
 (1)  015334  020103  047111  042524
 (1)  015342  051122  050125  020124
 (1)  015350  020055  044504  041523
 (1)  015356  047117  042516  052103
 (1)  015364  046040  041524  000040
7198          000001                    .END
```

```
ABASE = 174600          6060#   6073    6176    6177    6178    6179
ACDW1 = 000000          6073
ACDW2 = 000000          6073
ACPUOP= 000000          6073
ADDW0 = 000000          6073
ADDW1 = 000000          6073
ADDW10= 000000          6073
ADDW11= 000000          6073
ADDW12= 000000          6073
ADDW13= 000000          6073
ADDW14= 000000          6073
ADDW15= 000000          6073
ADDW2 = 000000          6073
ADDW3 = 000000          6073
ADDW4 = 000000          6073
ADDW5 = 000000          6073
ADDW6 = 000000          6073
ADDW7 = 000000          6073
ADDW8 = 000000          6073
ADDW9 = 000000          6073
ADEVCT= 000000          6073
ADEVM = 000001          6062#   6073
AENV  = 000000          6073
AENVM = 000000          6073
AFATAL= 000000          6073
ALTERN  007452          6882    6923#
AMADR1= 000000          6073
AMADR2= 000000          6073
AMADR3= 000000          6073
AMADR4= 000000          6073
AMAMS1= 000000          6073
AMAMS2= 000000          6073
AMAMS3= 000000          6073
AMAMS4= 000000          6073
AMSGAD= 000000          6073
AMSGLG= 000000          6073
AMSGTY= 000000          6073
AMTYP1= 000000          6073
AMTYP2= 000000          6073
AMTYP3= 000000          6073
AMTYP4= 000000          6073
APASS = 000000          6073
APRIOR= 000000          6073
APTCSU= 000040          7135    7136#
APTENV= 000001          7135    7136#   7139
APTSIZ= 000200          6197    7136#
APTSPO= 000100          7135    7136#
ASWREG= 000000          6073
ATESTN= 000000          6073
AUNIT = 000000          6073
AUSWR = 000000          6073
AVECT1= 000210          6061#   6073    6183    6184
AVECT2= 000000          6073
BIT0  = 000001          6059#
BIT00 = 000001          6059#   6293    6298    6347    7027
BIT01 = 000002          6059#
```

```
BIT02 = 000004          6059#
BIT03 = 000010          6059#
BIT04 = 000020          6059#
BIT05 = 000040          6059#
BIT06 = 000100          6059#
BIT07 = 000200          6059#   6222
BIT08 = 000400          6059#   7149
BIT09 = 001000          6059#   7139    7149
BIT1  = 000002          6059#
BIT10 = 002000          6059#   7139
BIT11 = 004000          6059#   7149
BIT12 = 010000          6059#
BIT13 = 020000          6059#   6738    7139
BIT14 = 040000          6059#   7149
BIT15 = 100000          6059#
BIT2  = 000004          6059#
BIT3  = 000010          6059#
BIT4  = 000020          6059#
BIT5  = 000040          6059#
BIT6  = 000100          6059#   6429
BIT7  = 000200          6059#
BIT8  = 000400          6059#
BIT9  = 001000          6059#
BPTVEC= 000014          6059#
BRKVEC= 000140          6059#
CKDAT   010460          6642    7089#
CKDAT1  010526          6744    6835    6873    7108#
CKSTAT  010156          6513    6556    6598    6632    6726    6825    6863    6899    6941    7009#
CKSWR = 104407          7139    7146    7149    7155#
CLKMES  015326          7197#
CORSZ   001540          6190#   6223    6234*   6845
CORSZR  002304          6224    6229#
CR    = 000015          6059#   7135
CRLF  = 000200          6059#   6211    7135
DBUF    015316          6191    6236    6501    6502*   6522    6545    6566    6568    6587    6607    6620    6653    6682
                        6714    6810    6878    6916    6958    7043    7047    7052    7089    7096    7192#
DBUFP   001542          6191#   6236*   6843*   6844*   6845    6848    6878*   6886*   6916*   6928*   6958*   7025    7066
                        7109
DDISP = 177570          6059#   6073    6197
DH1     014757          6076    6082    6088    6094    6100    6106    6112    6118    6124    6154    6160    7176#
DH2     015024          6130    6136    6169    7177#
DH3     015101          6142    6148    7178#
DISPLA  001142          6073#   6197*   7139*   7149*
DISPRE  000174          6065#   6197
DMAP    001536          6189#   6238*   6239*   6241    6966*   6975
DRVBAR  001522          6177#   6288    6291*   6292    6339*   6346    6349    6463    6465*   6466    6471    6501*   6545*
                        6587*   6620*   6653*   6682*   6714*   6755*   6781    6785    6810*   6848*   6887*   6929*   6969*
                        7026    7030
DRVCSR  001524          6178#   6361    6365*   6366    6376    6382    6384*   6386    6392    6407    6410*   6411    6424
                        6429*   6431    6435    6439    6447    6449*   6450    6454*   6456    6462*   6470*   6475*   6481*
                        6487*   6503*   6504*   6506    6508    6509*   6547*   6548*   6550    6552    6589*   6590*   6591
                        6593    6594*   6623*   6624*   6625    6627    6628*   6647    6655*   6656*   6663    6664*   6676
                        6684*   6685*   6690    6691*   6700    6717*   6718*   6719    6721    6722*   6757*   6758*   6759
                        6761    6762*   6766    6770    6790*   6791    6795    6799    6803    6813*   6814*   6818    6820
                        6821*   6851*   6852*   6856    6858    6859*   6890*   6891*   6892    6894    6895*   6932*   6933*
                        6934    6936    6937*   6970*   6997*   7009    7010*   7014
```

```
DRVCTO  001530          6183#   6204    6423*   6428*   6972*   6989*   6998*
DRVCT2  001532          6184#   6209    6973*   6990*   6998    6999*
DRVDBR  001526          6179#   6202    6308    6311*   6312    6324    6340*   6351    6353    6478    6480*   6482*   6483
                        6489    6524    6527    6546*   6571    6609    6612    6621*   6889*   6910    6913    6931*   6952
                        6955    6971*   7092    7123
DRVWCR  001520          6176#   6198    6249    6257    6271    6274*   6275    6338*   6342    6344    6500*   6544*   6586*
                        6619*   6652*   6681*   6715*   6716*   6754*   6774    6778    6811*   6849*   6885*   6927*   6968*
                        7017    7019
DSWR    = 177570        6059#   6073    6197
DT1     015130          6077    6083    6089    6095    6101    6107    6113    6119    6125    6155    6161    7181#
DT2     015144          6131    6137    6170    7182#
DT3     015162          6143    6149    7183#
EMTVEC= 000030          6059#   6197*
EM1     014066          6075    7158#
EM10    014277          6117    7165#
EM11    014325          6123    7166#
EM12    014354          6129    7167#
EM13    014375          6135    7168#
EM14    014414          6141    7169#
EM15    014462          6147    7170#
EM16    014650          6153    7173#
EM17    014665          6159    7174#
EM2     014105          6081    7159#
EM20    014722          6168    7175#
EM3     014127          6087    7160#
EM4     014144          6093    7161#
EM5     014206          6099    7162#
EM6     014231          6105    7163#
EM7     014255          6111    7164#
ERRVEC= 000004          6059#   6197*   6230*   6235*   6256*   6267*   7149*
GNS     = ****** U      6065    6211    7155
GTSWR   = 104406        6211    7155#
HT      = 000011        6059#   7135
IOADR   015264          7186#
IOPAGE  001544          6192#   6923    6928    6929    6951    6956
IOTEST  015230          7185#
IOTVEC= 000020          6059#   6197*
KDF     001546          6193#   6218*   6219    6220
KDF11B  015172          6216    7184#
LDBUF   010336          6585    7042#
LDBUF1  010410          6710    6809    6847    7066#
LF      = 000012        6059#   7135
LKVEC   = 000100        6059#
MTPS    = 106427        6063#
NXDEV   007672          6918    6924    6964#
NXDEV1  007674          6243    6965#   6976
ODTST   = 170000        6059#
PIRQ    = 177772        6059#
PIRQVE= 000240          6059#
POINT   = 015320        7197#
PR0     = 000000        6059#
PR1     = 000040        6059#
PR2     = 000100        6059#
PR3     = 000140        6059#
PR4     = 000200        6059#
PR5     = 000240        6059#
```

```
PR6    = 000300          6059#   6197    7152
PR7    = 000340          6059#
PS     = 177776          6059#
PSW    = 177776          6059#
PWRMSG  013752           7152    7153#
PWRVEC= 000024           6059#   6197*   7152*
RDCHR  = 104410          6217    7150    7155#
RDLIN  = 104411          7151    7155#
RDOCT  = 104412          7155#
RESTRT  002406           6240    6242    6244#   6977    7152
RESVEC= 000010           6059#
RSTVEC  010132           6379    6443    6536    6580    6614    6644    6673    6705    6748    6805    6839    6879    6917
                         6959    6997#
S       013600           7151#
SETUP2  002022           6200#   6203
SETUP3  002052           6207#   6210
SETVEC  010112           6358    6496    6540    6583    6617    6648    6677    6708    6752    6807    6841    6883    6925
                         6988#
STACK  = 001100          6059#   6197    6245
START   001550           6065    6197#
START1  002012           6198#   6978
STKLMT= 177774           6059#
SWR     001140           6073#   6197*   6211    6222*   6881    7139    7149    7150*   7152*
SWRCK   012222           7139    7144#
SWREG   000176           6065#   6197    6211    7150
SW0    = 000001          6059#
SW00   = 000001          6059#
SW01   = 000002          6059#
SW02   = 000004          6059#
SW03   = 000010          6059#
SW04   = 000020          6059#
SW05   = 000040          6059#
SW06   = 000100          6059#
SW07   = 000200          6059#
SW08   = 000400          6059#
SW09   = 001000          6059#
SW1    = 000002          6059#
SW10   = 002000          6059#
SW11   = 004000          6059#
SW12   = 010000          6059#
SW13   = 020000          6059#
SW14   = 040000          6059#
SW15   = 100000          6059#
SW2    = 000004          6059#
SW3    = 000010          6059#
SW4    = 000020          6059#
SW5    = 000040          6059#
SW6    = 000100          6059#
SW7    = 000200          6059#
SW8    = 000400          6059#
SW9    = 001000          6059#
TBITVE= 000014           6059#
TKVEC  = 000060          6059#
TMAIN   000000           6064#
TPVEC  = 000064          6059#
TRAPVE= 000034           6059#   6197*
```

```
TRTVEC= 000014            6059#
TSTNUM  001534            6188#   7144*   7181    7182    7183
TST1    002442            6251#
TST10   003442            6381#
TST11   003516            6388    6391#
TST12   003600            6405#
TST13   003676            6420#
TST14   004036            6445#
TST15   004140            6458    6461#
TST16   004242            6477#
TST17   004356            6491    6494#
TST2    002552            6269#
TST20   004606            6538#
TST21   005030            6582#
TST22   005236            6616#
TST23   005412            6646#
TST24   005566            6675#
TST25   005770            6707#
TST26   006226            6750#
TST27   006606            6806#
TST3    002650            6286#
TST30   007000            6840#
TST31   007224            6880#
TST4    002766            6302    6306#
TST5    003064            6323#
TST6    003144            6333    6336#
TST7    003272            6352    6356#
TYPDS = 104405            6978    7155#
TYPE  = 104401            6211    6216    6219    6225    6978    7135    7137    7138    7139    7148    7150    7151    7152
                          7155#   7197
TYPOC = 104402            7148    7150    7155#
TYPON = 104404            7155#
TYPOS = 104403            7155#
WARN    014521            6225    7171#
$APTHD  001000            6072#
$ASTAT= ****** U          7136
$ATYC   011136            7136#
$ATY1   011112            7136#
$ATY3   011120            7135    7136#
$ATY4   011130            7136#   7139
$AUTOB  001134            6073#   6211*   6214    7150
$BASE   001250            6073#   6199
$BDADR  001122            6073#   6259*   6271*   6288*   6308*   6325*   6344*   6349*   6353*   6361*   6382*   6393*   6407*
                          6424*   6447*   6463*   6478*   6506*   6527*   6550*   6571*   6591*   6612*   6625*   6647*   6676*
                          6719*   6759*   6770*   6778*   6785*   6795*   6803*   6818*   6856*   6892*   6913*   6934*   6955*
                          7014*   7019*   7030*   7092*   7123*   7181    7182    7183
$BDDAT  001126            6073#   6255*   6275*   6276    6292*   6293*   6294    6312*   6313    6328*   6329    6330    6331*
                          6332    6342*   6346*   6347*   6351*   6366*   6367*   6368    6376*   6377*   6386*   6387    6396*
                          6397    6398    6399*   6400    6411*   6412*   6413    6431*   6435*   6439*   6440    6450*   6451
                          6456*   6457    6466*   6467    6471*   6483*   6484    6489*   6490    6508*   6524*   6525    6552*
                          6568*   6569    6593*   6609*   6610    6627*   6651*   6660*   6663*   6670    6680*   6687*   6690*
                          6697    6700*   6702    6721*   6761*   6766*   6768    6774*   6776    6781*   6783    6791*   6793
                          6799*   6801    6815*   6816*   6820*   6853*   6854*   6858*   6894*   6910*   6911    6936*   6952*
                          6953    7009*   7012    7017*   7026*   7027*   7028    7093*   7120*   7181    7182
$BELL   001164            6073#   7139
$CDW1   001254            6073#
```

```
$CDW2   001256          6073#
$CHARC  011106          7135#*
$CKSWR  012664          7150#   7155
$CLKVE  015320          7197#
$CMTAG  001100          6073#   6197
$CM3  =  000000         6073#
$CNTLG  013411          7150#
$CNTLU  013404          7150#
$CPUOP  001222          6073#
$CRLF   001171          6073#   7135    7139    7148    7150    7151
$DBLK   012022          7138#
$DDW0   001260          6073#   6249*
$DDW1   001262          6073#
$DDW10  001304          6073#
$DDW11  001306          6073#
$DDW12  001310          6073#
$DDW13  001312          6073#
$DDW14  001314          6073#
$DDW15  001316          6073#
$DDW2   001264          6073#
$DDW3   001266          6073#
$DDW4   001270          6073#
$DDW5   001272          6073#
$DDW6   001274          6073#
$DDW7   001276          6073#
$DDW8   001300          6073#
$DDW9   001302          6073#
$DEVCT  001204          6073#   6246*
$DEVM   001252          6073#   6238
$DOAGN  010066          6978#
$DTBL   012012          7138#
$ENDAD  010056          6070    6978#
$ENDCT  010024          6978#
$ENDMG  010075          6978#
$ENULL  010072          6978#
$ENV    001214          6073#   6211    7135    7136    7139
$ENVM   001215          6073#   6197    7135    7136
$EOP    007770          6967    6978#
$EOPCT  010016          6978#
$ERFLG  001103          6073#   7139*   7149*
$ERMAX  001115          6073#   6197*   7149*
$ERROR  012032          6197    7139#
$ERRPC  001116          6073#   7139*   7148    7181    7182    7183
$ERRTB  001320          6073#   7148
$ERRTY  012240          7145    7148#
$ERTTL  001112          6073#   7139*
$ESCAP  001162          6073#   6197*   7139    7149*
$ETABL  001214          6073#
$ETEND  001320          6072    6073#
$FATAL  001176          6073#   7136*
$FFLG   011356          7136#*
$FILLC  001156          6073#   7135
$FILLS  001155          6073#   7135
$GDADR  001120          6073#   6522*   6566*   6607*   6914*   6956*   7094*   7121*   7182
$GDDAT  001124          6073#   6254*   6273*   6274    6276    6279*   6282*   6283*   6290*   6291    6294    6297*   6298*
                        6301*   6303*   6310*   6311    6313    6316*   6319*   6320*   6327*   6332    6337*   6363*   6364*
```

```
                    6368    6375*   6383*   6387    6395*   6400    6408*   6413    6416*   6425*   6440    6448*   6451
                    6455*   6457    6464*   6467    6473*   6479*   6480    6484    6488*   6490    6507*   6523*   6525
                    6551*   6567*   6569    6592*   6608*   6610    6626*   6662*   6689*   6701*   6702    6720*   6760*
                    6767*   6768    6775*   6776    6782*   6783    6792*   6793    6800*   6801    6819*   6857*   6893*
                    6909*   6911    6935*   6951*   6953    7011*   7012    7020*   7023*   7024*   7025*   7028    7122*
                    7181    7182
$GET42  010046      6978#
$GTSWR  012734      7150#   7155
$HD   = 000003      6054
$HIBTS  001000      6072#
$HIOCT  013576      7151#*
$ICNT   001104      6073#   7149*
$ILLUP  013744      7152#
$INTAG  001135      6073#   7150
$ITEMB  001114      6073#   7139*   7148
$LF     001172      6073#   7135    7139    7150    7151
$LFLG   011355      7136#*
$LPADR  001106      6073#   6197*   6251*   7149*
$LPERR  001110      6073#   6197*   6253*   6270*   6287*   6307*   6360*   6406*   6495*   6539*   7139    7149*
$MADR1  001226      6073#
$MADR2  001232      6073#
$MADR3  001236      6073#
$MADR4  001242      6073#
$MAIL   001174      6072    6073#   6197    6211    6251    7135    7139    7149
$MAMS1  001224      6073#
$MAMS2  001230      6073#
$MAMS3  001234      6073#
$MAMS4  001240      6073#
$MBADR  001002      6072#
$MFLG   011354      7136#*
$MNEW   013427      7150#
$MSGAD  001210      6073#   7136*
$MSGLG  001212      6073#   7136*
$MSGTY  001174      6073#   7136*
$MSWR   013416      7150#
$MTYP1  001225      6073#
$MTYP2  001231      6073#
$MTYP3  001235      6073#
$MTYP4  001241      6073#
$MXCNT  012662      7149#
$NULL   001154      6073#   7135
$NWTST= 000001      6251#   6269#   6286#   6306#   6323#   6336#   6356#   6381#   6391#   6405#   6420#   6445#   6461#
                    6477#   6494#   6538#   6582#   6616#   6646#   6675#   6707#   6750#   6806#   6840#   6880#
$OCNT   011602      7137#*
$OMODE  011604      7137#*
$OVER   012646      7149#
$PASS   001202      6073#   6197*   6978*   7149
$PASTM  001006      6072#
$PWRAD  013740      7152#
$PWRDN  013600      6197    7152#
$PWRMG  013734      7152#
$PWRUP  013652      7152#
$QUES   001170      6073#   7135    7139    7150    7151
$RDCHR  013146      7150#   7155
$RDDEC= ****** U    7155
$RDLIN  013266      7150#   7155
```

```
$RDOCT  013440          7151#   7155
$RDSZ = 000010          7150#
$RTNAD  010070          6978#
$R2A  = ****** U        7155
$SAVRE= ****** U        7155
$SAVR6  013750          7152#*
$SCOPE  012374          6197    7149#
$SETUP= 000117          6195#   6197    6211    6978    7139    7149    7150
$STUP = 177777          6195#
$SVLAD  012612          7149#
$SVPC = 000106          6070#
$SWR  = 167400          6054#   6055#   6058    6073    6197    6251    6269    6286    6306    6323    6336    6356    6381
                        6391    6405    6420    6445    6461    6477    6494    6538    6582    6616    6646    6675    6707
                        6750    6806    6840    6880    6978    7139    7149    7152
$SWREG  001216          6073#   6197
$SWRMK= 000300          6056#   6058    7149
$TESTN  001200          6073#   6251*   7149*
$TIMES  001160          6073#   6197*   6336*   6356*   6381*   6420*   6582*   6616*   6646*   6675*   6707*   6750*   6806*
                        6840*   6978*   7149*
$TKB    001146          6073#   7150
$TKS    001144          6073#   7150*
$TN   = 000032          6054#   6057#   6251#   6269#   6286#   6302    6306#   6323#   6333    6336#   6352    6356#   6381#
                        6388    6391#   6405#   6420#   6445#   6458    6461#   6477#   6491    6494#   6538#   6582#   6616#
                        6646#   6675#   6707#   6750#   6806#   6840#   6880#
$TPB    001152          6073#   7135*
$TPFLG  001157          6073#   7135
$TPS    001150          6073#   6886    6887    6909    6914    7135
$TRAP   014004          6197    7155#
$TRAP2  014026          7155#
$TRP  = 000013          7155#
$TRPAD  014040          7155#
$TSTM   001004          6072#
$TSTNM  001102          6073#   6252*   6978*   7139    7144    7149*
$TTYIN  013374          7150#
$TYPBN= ****** U        7155
$TYPDS  011606          7138#   7155
$TYPE   010630          7135#   7136    7155
$TYPEC  011042          7135#   7150
$TYPEX  011110          7135#
$TYPOC  011404          7137#   7155
$TYPON  011420          7137#   7155
$TYPOS  011360          7137#   7155
$UNIT   001206          6073#   6237*   6246    6247    6974*
$UNITM  001010          6072#
$USWR   001220          6073#
$VECT1  001244          6073#   6205
$VECT2  001246          6073#
$XTSTR  012410          7149#
$$GET4= 000000          6978#
$OFILL  011603          7137#*
$40CAT= ****** U        7139    7149
.     = 015372          6065#   6066#   6070#   6071#   6072#   6073#   6197    6211#   6978    7135    7136#   7138#   7139
                        7148#   7149    7150#   7151    7152    7187#   7197#
.$ASTA= ****** U        7136
.$X   = 001000          6072#
```

```
COMMEN  1566#  6059#
ENDCOM  1578#  6059#
ERROR   6059#  6266  6278  6296  6315  6334  6345  6350  6354  6370  6378  6389  6402  6415  6432
        6436   6442  6453  6459  6469  6474  6486  6492  6510  6516  6518  6520  6528  6553  6559
        6562   6564  6572  6595  6601  6603  6605  6613  6629  6635  6638  6640  6643  6665  6672
        6692   6699  6704  6723  6729  6733  6735  6746  6763  6771  6779  6786  6796  6804  6822
        6828   6831  6833  6837  6860  6866  6869  6871  6875  6896  6902  6905  6907  6915  6938
        6944   6947  6949  6957
ESCAPE  1698#  6059#
GETPRI  1312#  6059#
GETSWR  1771#  6059#  6211#
MULT    4484#  6059#
NEWTST  1626#  6059#  6251  6269  6286  6306  6323  6336  6356  6381  6391  6405  6420  6445  6461
        6477   6494   6538  6582  6616  6646  6675  6707  6750  6806  6840  6880
OK      6163#  6276  6294  6313  6332  6368  6387  6400  6413  6440  6451  6457  6467  6484  6490
        6525   6569  6610  6702  6768  6776  6783  6793  6801  6911  6953  7012  7028
POP     2153#  6059#  7136  7138  7151  7152
PUSH    2145#  6059#  7136  7138  7151  7152
REPORT  5463#  6059#
SCOPE   6059#  6269  6286  6306  6323  6336  6356  6381  6391  6405  6420  6445  6461  6477  6494
        6538   6582  6616  6646  6675  6707  6750  6806  6840  6880  6964
SETPRI  1279#  6059#
SETTRA  7155#
SETUP   1337#  6059#  6197
SKIP    1733#  6059#  6302  6333  6352  6388  6458  6491
SLASH   1517#  6059#
SPACE   6059#
STARS   1485#  6059#  6070  6072  6073  6226  6228  6251  6269  6286  6306  6323  6336  6356  6381
        6391   6405   6420  6445  6461  6477  6494  6538  6582  6616  6646  6675  6707  6750  6806
        6840   6880   6920  6922  6961  6963  6978  6983  6987  6993  6996  7003  7008  7036  7041
        7060   7065   7084  7088  7101  7107  7135  7136  7137  7138  7139  7140  7143  7148  7150
        7151   7152   7155  7188  7191
SWRSU   1453#  6059#  6197#
TRMTRP  7155#
TYPBIN  2088#  6059#
TYPDEC  2058#  6059#  6978
TYPNAM  1826#  6059#  6211
TYPNUM  2025#  6059#
TYPOCS  1978#  6050#  6059#
TYPOCT  1941#  6059#  7148  7150
TYPTXT  1894#  6059#
$$CMRE  6073#
$$CMTM  6073#
$$ESCA  1711#  6059#
$$NEWT  1662#  6059#  6251  6269  6286  6306  6323  6336  6356  6381  6391  6405  6420  6445  6461
        6477   6494   6538  6582  6616  6646  6675  6707  6750  6806  6840  6880
$$SET   7155#
$$SETM  6197#
$$SKIP  1746#  6059#  6302  6333  6352  6388  6458  6491
.EQUAT   189#  6050#  6059
.HEADE    61#  6049#  6054
.INIT   5658#  6059#  7197
.SETUP  1213#  6049#  6195
.SWRHI   104#  6050#  6058
.SWRLO  6058#
.$ACT1  5064#  6052#  6070
```

```
.$APTB   5109#   6052#   6073#
.$APTH   5370#   6052#   6072
.$APTY   5547#   6052#   7136
.$ASTA   5417#
.$CATC    932#   6050#   6065
.$CMTA   1047#   6050#   6073
.$DB2D   4686#
.$DB2O   4812#
.$DIV    4587#
.$EOP    2214#   6051#   6978
.$ERRO   2700#   6051#   7139
.$ERRT   2896#   6051#   7148
.$MULT   4523#
.$POWE   4229#   6050#   7152
.$RAND   4307#
.$RDDE   3891#
.$RDOC   3797#   6049#   7151
.$READ   3395#   6051#   7150
.$R2AZ   4958#
.$SAVE   3969#
.$SB2D   4771#
.$SB2O   4874#
.$SCOP   2454#   6051#   7149
.$SIZE   4361#
.$SUPR   4913#
.$TRAP   4073#   6049#   7155
.$TYPB   3287#
.$TYPD   3209#   6051#   7138
.$TYPE   2985#   6051#   7135
.$TYPO   3112#   6050#   7137
.$40CA    972#

. ABS.  015372     000

 ERRORS DETECTED:  0

 CNDRAA,CNDRAA/CRF/NL:TOC=CNMAC2.SML,CNDRAA.P11
 RUN-TIME: 13 13 .9 SECONDS
 RUN-TIME RATIO: 53/28=1.8
 CORE USED:  33K  (66 PAGES)
```