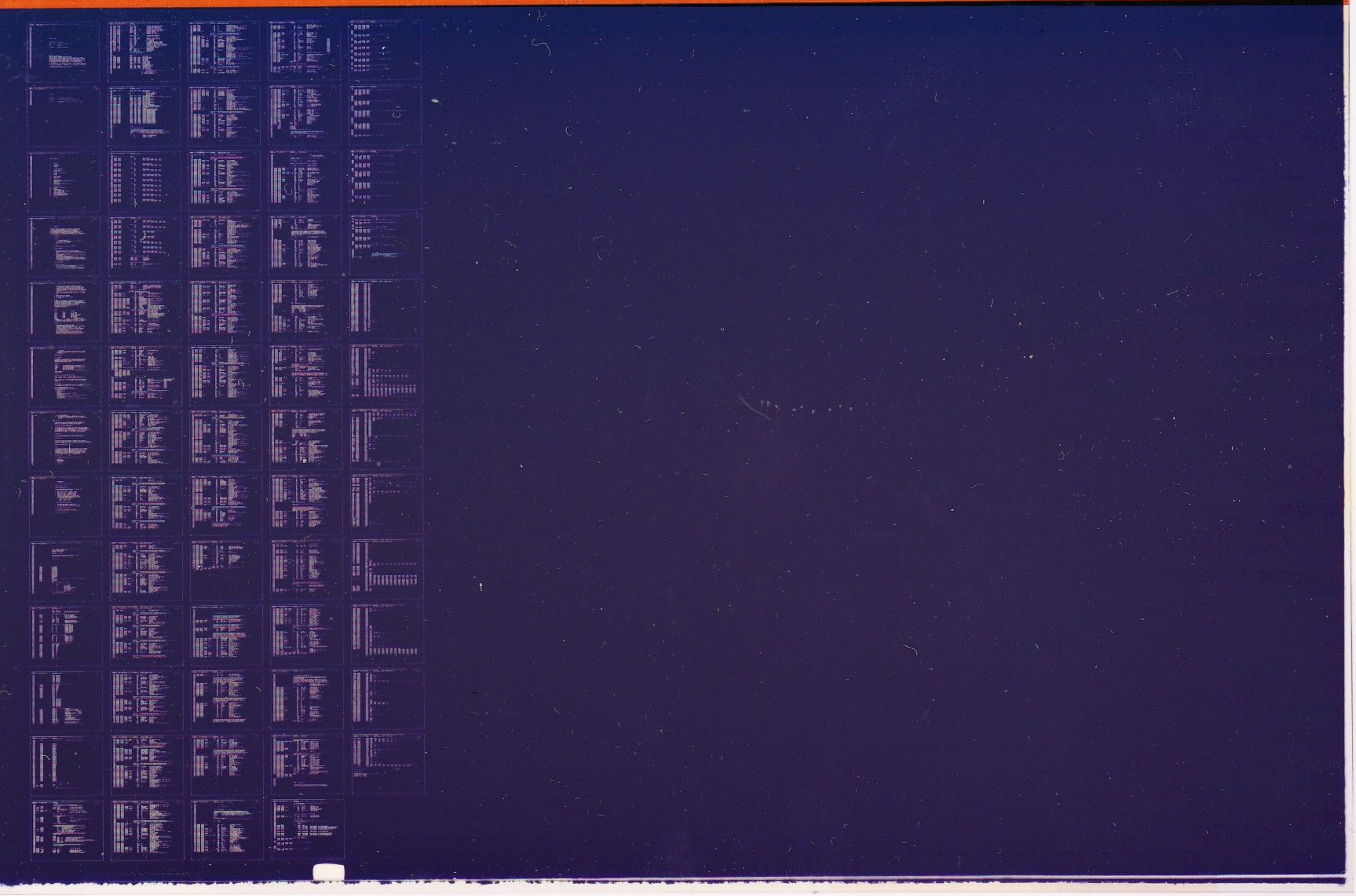
DRV11-B

DRV11-B DMA INTFC
CVDRAC0

AH-T086C-MC
1 OF 1 OCT 1985

digital
MADE IN USA



```
5624
5625
5626                                    .REM ¤
5627
5628
5629
5630
5631
5632
5633
5634
5635
5636
5637                            IDENTIFICATION
5638                            --------------
5639
5640
5641                            PRODUCT CODE:   AC-8178C-MC
5642
5643                            PRODUCT NAME:   CVDRACO DRV11B DMA INTFC DIAG
5644
5645                            DATE:           JUNE 1985
5646
5647                            MAINTAINER:     DIAGNOSTIC ENGINEERING
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657                            COPYRIGHT (C) 1976,1985
5658                            DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
5659                            THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
5660                            COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE
5661                            COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT
5662                            BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR
5663                            USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS.  TITLE
5664                            TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
5665
5666                            THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND
5667                            SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
5668
5669                            DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE
5670                            ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
5671
```

```
5673
5674
5675
5676
5677                            HISTORY
5678                            -------
5679
5680
5681                            REVISION A      :FIRST RELEASE OF DIAGNOSTIC
5682
5683                            REVISION B      :THIS REVISION WAS MADE TO SUPPORT 11/23-B
5684
5685                            REVISION C      :THIS REVISION WAS MADE TO SUPPORT ORION (11/73)CPU
5686                                             SEE ;DD001
5687
5688
5689
```

TABLE OF CONTENTS

```
5741
5742
5743
5744
5745
5746
5747                    1.0     ABSTRACT
5748                    ----------------
5749
5750                    THE DRV11B DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED
5751                    TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE WITH
5752                    THE LOOP BACK CABLE INSERTED IN THE USER I/O CONNECTORS.
5753                    TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL
5754                    VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5
5755                    OF THIS DOCUMENT. IF THE SYSTEM ALSO INCLUDES AN "REV11" (DMA REFRESH),
5756                    THE DMA REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.
5757
5758                    2.0     REQUIREMENTS
5759                            ------------
5760
5761                    2.1     EQUIPMENT
5762
5763                            1.  LSI-11 FAMILY PROCESSOR.
5764                                11/03(LSI-11/02), 11/23(KDF11-A), 11/23B(KDF11-B)
5765                            2.  DLV11 WITH I/O TYPE TERMINAL
5766                            3.  DRV11B WITH LOOP BACK CABLE
5767
5768                    2.2     STORAGE
5769
5770                            THE PROGRAM USES THE LOWER 4K OF MEMORY.
5771
5772                    3.0     LOADING PROCEDURE
5773                            ----------------
5774
5775                            IF USING PAPER TAPE READER FOLLOW THIS PROCEDURE:
5776                            1.  ASSURE THAT THE LSI-11 IS IN THE ODT MICROCODE STATE.
5777                            2.  LOAD THE LOW OR HIGH SPEED READER WITH THE ABSOLUTE LOADER
5778                                TAPE.
5779                            3.  TYPE THE READER'S CSR ADDRESS (177560-LOW OR 177550-HIGH)
5780                                AND CHARACTER 'L'.
5781                            4.  AFTER TAPE IS LOADED, LOAD THE DRV11B BINARY TAPE INTO THE
5782                                READER AND TYPE THE CHARACTER 'P'.
5783                            5.  IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STEPS 2 & 3),
5784                                THEN ONLY THE STARTING ADDRESS OF THE ABSOLUTE LOADER AND
5785                                THE CHARACTER 'G' NEED BE TYPED (WITH THE DRV11B BINARY TAPE
5786                                IN THE APPROPRIATE READER).
5787
5788                    4.0     STARTING PROCEDURE
5789                            ------------------
5790
5791                            FOR PAPER TAPE MEDIA:
5792                            1.  MAKE SURE THE MAINTENANCE LOOP BACK CABLE IS INSERTED IN
5793                                THE I/O CONNECTORS ON THE M7950 MODULE.
5794                            2.  MAKE SURE THE DEVICE BUS & VECTOR ADDRESSES AGREE WITH THE
5795                                DEFAULT VALUES DEFINED IN SECTION 7.1.  IF NOT, CHANGE
5796                                LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
```

```
5797                            3.  INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
5798                            4.  TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
5799                            5.  THE PROGRAM WILL RESPOND BY TYPING THE SOFTWARE SWITCH
5800                                REGISTER CONTENTS AND ALLOWING THE USER TO CHANGE ITS
5801                                CONTENTS BY ENTERING OCTAL SWITCH REGISTER DATA TERMINATED
5802                                BY A CARRIAGE RETURN - SEE SECTION 5.0 FOR SWITCH REGISTER
5803                                OPTIONS.
5804                            6.  THE NEXT QUESTION ASKED IS ABOUT THE TYPE OF A PROCESSOR.
5805                                TYPE 'Y' WITH CARRIGE RETURN IF USING KDF11-B, 'N' OTHER-
5806                                WISE.
5807
5808                            IF RUNNING UNDER XXDP+ MONITOR:
5809                            A)  DO 1., 2., 3. OF THE ABOVE.
5810                            B)  IN MONITOR MODE TYPE IN 'R VDRAB?'.
5811                            C)  DO 5. AND 6.
5812
5813                            *****
5814                            IF USING LSI-11/23B(KDF11-B) AND WANT TO TEST DMA TRANSFERS TO
5815                            I/O PAGE PUT THE ADDRESS OF YOUR I/O INTERFACE CONTROL RE-
5816                            GISTER INTO LOCATION 1544 (IOPAGE). TO DO THIS PERMANENTLY USE
5817                            LOAD-MOD-DUMP PROCEDURE DEFINED IN 7.2. FOR TEMPORARY CHANGES
5818                            JUST MODIFY LOCATIONS AFTER LOADING THE PROGRAM.
5819                            NOTE: THIS TEST IS NOT GOING TO BE PERFOMED UNLESS THE ABOVE
5820                            MENTIONED LOCATIONS ARE MODIFIED.
5821                    5.0     SOFTWARE SWITCH REGISTER
5822                            ------------------------
5823
5824                    5.1     OPTIONS
5825
5826                            SWITCH          OCTAL           FUNCTION
5827                            ------          -----           --------
5828                            SW15=1          100000          HALT ON ERROR
5829                            SW14=1          040000          LOOP ON TEST
5830                            SW13=1          020000          INHIBIT ERROR TYPEOUTS
5831                            SW11=1          004000          INHIBIT ITERATIONS
5832                            SW10=1          002000          BELL ON ERROR
5833                            SW09=1          001000          LOOP ON ERROR
5834                            SW08=1          0004XX          LOOP ON TEST IN SWR <5-0>
5835                            ***SW07=1               000200          DEVICE IS KDF11-B
5836                            ***DIAGNOSTIC WILL SET THIS SWITCH AUTOMATICALLY, IF THIS TYPE
5837                            OF A PROCESSOR HAS BEEN CONFIRMED IN A DIALOGUE. IF RUNNING IN
5838                            AUTOMATIC MODE (CHAINS UNDER XXDP OR APT) SEE SECTION 7.2
5839
5840                    5.2     CONTROL
5841
5842                            1.  THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE
5843                                CHANGED BY USING THE ODT FACILITIES.
5844                            2.  THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM
5845                                CONTROL BY TYPING THE 'CONTROL & G' KEYS.  THIS
5846                                KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS
5847                                AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED
5848                                WITH A CARRIAGE RETURN.
5849                            3.  ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN
5850                                ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2
5851                                ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE
5852                                SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING
```

```
5853                                    'P' (CONTINUE).
5854                                4.  IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL
5855                                    'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE
5856                                    BY THE PROGRAM.
5857
5858                        6.0     ERROR REPORTING
5859                                --------------
5860
5861                        6.1     ERROR COMMENT
5862
5863                                ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE
5864                                COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF
5865                                THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE
5866                                ERROR PC OR FROM THE TEST ITSELF.
5867
5868                        6.2     ERROR DATA
5869
5870                                *ERRPC          LISTING ADDRESS WHERE THE ERROR WAS DETECTED
5871                                *TSTNUM         TEST NUMBER WHERE THE ERROR OCCURRED
5872                                 BUSADR         DRV11B BUS REG ADDRESS OF CONCERNED OPERATION
5873                                 EXPCT          DATA THAT WAS EXPECTED
5874                                 RCVD           DATA THAT WAS RECEIVED
5875                                 ADRS           MEMORY ADDRESS OF DATA TRANSFER ON ERROR
5876
5877                                *ALWAYS REPORTED
5878                        7.0     MISCELLANEOUS
5879                                -------------
5880
5881                        7.1     DRV11B BUS & VECTOR ADDRESS MODIFICATION
5882
5883                                MODIFY LOCATION '$BASE' IF BASE BUS ADDRESS IS NOT 172410.
5884                                MODIFY LOCATION '$VECT1' IF VECTOR ADDRESS IS NOT 124.
5885
5886                                *NOTE:  USE THE LSI-11 ODT FACILITIES TO MODIFY THESE LOCATIONS
5887                                AFTER PROGRAM LOAD.  NO VECTOR ASSIGNMENT ABOVE 774 SHOULD BE
5888                                ALLOWED.
5889
5890                        7.2     XXDP/APT NOTES
5891
5892                                THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REF. 7.5)(REQUIRES 8K OR MORE).
5893                                THIS DIAGNOSTIC DOES SUPPORT "APT" AND HAS RUN UNDER IT.
5894
5895                                IF THE PROCESSOR USED IS KDF11-B:
5896                                FOR APT SET $SWREG BIT07 TO 1 (000200)
5897                                FOR XXDP CHAINS SET LOCATION 176 (SWR) BIT07 TO 1
5898                                TO DO THIS UNDER XXDP:
5899                                  1. R UPD2
5900                                  2. LOAD VDRAB?.BIC
5901                                  3. MOD 176
5902                                     THE TERMINAL WILL RESPOND: 176/000000
5903                                  4. NOW TYPE IN 200
5904                                  5. BE CAREFUL: AT THIS POINT IT IS NECESARY TO DELETE THE FILE
5905                                     FROM THE DISK.
5906                                     DEL DL0:VDRA8?.BIC (IF MEDIA IS RL ON DRIVE 0)
5907                                  6. DUMP VDRAB0.BIC
5908                                     TO BE SAFE, SKIP 5. AND DUMP THE FILE UNDER DIFFERENT NAME
```

```
5909                                    WITH .BIC EXTENSION.
5910                                NOTE: THIS PROCEDURE ASSUMES THAT DIAGNOSTIC IS ON THE DISK
5911                                      FROM WHICH THE SYSTEM IS BOOTED. IF THIS IS UNTRUE
5912                                      IN 2. AND 6. THE OPERATOR HAVE TO SPECIFY THE DRIVE BEFORE
5913                                      THE NAME.
5914
5915                            7.3     POWER FAIL
5916
5917                                    A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
5918                                    WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
5919                                    NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).
5920
5921                            7.4     MULTIPLE DRV11B INTERFACE TESTING
5922
5923                                    THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11B'S CONNECTED.
5924                                    THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 DRV11B INTERFACES
5925                                    WITH CONTIGUOUS BUS AND VECTOR ADDRESSES.  THIS IS ACCOMPLISHED
5926                                    BY THE OPERATOR SETTING UP LOCATION '$DEVM' WITH A BIT MAP INDICATING WHAT
5927                                    INTERFACES ARE TO TESTED.  I.E. BIT0=1 SAYS TEST 1ST DRV11B,
5928                                    BIT1=1 SAYS TEST 2ND DRV11B, BIT2=1 SAYS TEST 3RD DRV11B, ETC..
5929
5930                            7.5     RESTRICTIONS
5931
5932                                    IF THE SYSTEM ALSO INCLUDES AN "REV11" (DMA REFRESH),  THE DMA
5933                                    REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.
5934
5935                            8.0     EXECUTION TIME
5936                                    --------------
5937
5938                                    EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
5939                                    TO ABOUT 90 SECONDS WITH ITERATIONS ENABLED WITH ONE DRV11B CONNECTED.
5940                                    AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.
5941
5942                            9.0     PROGRAM TEST DESCRIPTIONS
5943                                    ------------------------
5944
5945                            9.1     GENERAL
5946
5947                                    THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED
5948                                    TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV11B DMA INTERFACE.
5949                                    A HIGH DEGREE OF TESTING IS ACCOMPLISHED WITH THE AID OF THE
5950                                    MAINTENANCE LOOP BACK CABLE PROVIDED FOR DIAGNOSTIC TESTING.
5951                                    A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS
5952                                    AT THE BEGINNING OF THE LISTING.  THE COMMENT FIELD WITHIN
5953                                    EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.
5954
5955                            9.2     REGISTER TESTS
5956
5957                                    THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:
5958
5959                                    1.  WORD COUNT
5960                                    2.  BUFFER ADDRESS
5961                                    3.  COMMAND/STATUS
5962                                    4.  DATA BUFFER
5963
5964                            9.3     BYTE ADDRESSING TESTS
```

```
5965
5966                                    1.  COMMAND/STATUS
5967                                    2.  DATA BUFFER
5968
5969                            9.4     'FNCT' TO 'STAT' WRAP AROUND TEST
5970
5971                            9.5     READY INTERRUPT TEST
5972
5973                            9.6     NPR DATA TRANSFER TESTS
5974
5975                                    THE FOLLOWING NPR XFERS ARE CHECKED FOR CORRECT STATUS,
5976                                    WORD COUNT, BUFFER ADDRESS & DATA:
5977
5978                                    1.  SINGLE 'DATI' XFER - FLOATING I/O PTRN
5979                                    2.  SINGLE 'DATO' XFER - FLOATING I/O PTRN
5980                                    3.  200 'DATI' XFERS - FLOATING I/O PTRN
5981                                    4.  200 'DATO' XFERS - FLOATING I/O PTRN
5982                                    5.  SINGLE 'DATI' XFER TO THE TTY PRINTER CSR
5983                                        FOR KDF11-B PROCESSOR THE USER HAVE TO SELECT
5984                                        CSR OF A PARTICULAR INTERFACE (DISK INTERFACES
5985                                        ARE SUGGESTED). SOME OF THEM ARE:
5986                                            RXV11   (RX01): 177170
5987                                            RXV21   (RX02): 177170
5988                                            RKV11-D (RK05): 177404
5989
5990                            9.7     MAINT MODE NPR DATA TRANSFER TESTS
5991
5992                                    1.  THAT MAINT MODE CONTROLS 'FNCT' BITS
5993                                    2.  200 MAINT MODE XFERS - CHECKING STATUS & DATA
5994                                    3.  200 MAINT MODE XFERS TO EACH 4K AVAILABLE MEM
5995
5996                            9.8     BURST & NON-BURST MODE TESTS
5997
5998                                    1.  THAT CPU IS LOCKED OUT IN BURST MODE
5999                                    2.  THAT CPU IS NOT LOCKED OUT IN NON-BURST MODE
6000
6001                            9.9     'NEX' ERROR CONDITION TEST
6002
6003                            10.0    LISTING
6004                                            -------
```

```
                                .TITLE  MAINDEC-11-CVDRA-C DRV11B DMA INTERFACE DIAGNOSTIC
                                ;*COPYRIGHT (C) FEBRUARY 1985
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY R. MOORE
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), FEB, 1985.
                                ;*

                                .ENABLE ABS,AMA




000007                          MFPT=000007
104401                          TYPE=104401
104402                          TYPOC=104402
104403                          TYPOS=104403
104404                          TYPON=104404
104405                          TYPDS=104405
104406                          GTSWR=104406
104407                          CKSWR=104407
104410                          RDCHR=104410
104411                          RDLIN=104411
104412                          RDOCT=104412
000021                          $XON=000021
000023                          $XOFF=000023
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

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

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

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

```
                                .EQUIV  SW08,SW8
                                .EQUIV  SW07,SW7
                                .EQUIV  SW06,SW6
                                .EQUIV  SW05,SW5
                                .EQUIV  SW04,SW4
                                .EQUIV  SW03,SW3
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
        100000                  BIT15=  100000
        040000                  BIT14=  40000
        020000                  BIT13=  20000
        010000                  BIT12=  10000
        004000                  BIT11=  4000
        002000                  BIT10=  2000
        001000                  BIT09=  1000
        000400                  BIT08=  400
        000200                  BIT07=  200
        000100                  BIT06=  100
        000040                  BIT05=  40
        000020                  BIT04=  20
        000010                  BIT03=  10
        000004                  BIT02=  4
        000002                  BIT01=  2
        000001                  BIT00=  1
                                .EQUIV  BIT09,BIT9
                                .EQUIV  BIT08,BIT8
                                .EQUIV  BIT07,BIT7
                                .EQUIV  BIT06,BIT6
                                .EQUIV  BIT05,BIT5
                                .EQUIV  BIT04,BIT4
                                .EQUIV  BIT03,BIT3
                                .EQUIV  BIT02,BIT2
                                .EQUIV  BIT01,BIT1
                                .EQUIV  BIT00,BIT0

                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
        000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
        000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
        000014                  TBITVEC=14              ;;"T" BIT
        000014                  TRTVEC= 14              ;;TRACE TRAP
        000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
        000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
        000024                  PWRVEC= 24              ;;POWER FAIL
        000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
        000034                  TRAPVEC=34              ;;"TRAP" TRAP
        000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
        000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
        000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
        172410                  ABASE=  172410          ;BASE DRV11B BUS ADRS EQUATE
        104400                  TRAP=104400
        000124                  AVECT1= 000124          ;BASE DRV11B VECTOR ADRS EQUATE -
        000001                  ADEVM=  1               ;DEFAULT TO ONE DRV11B
        106427                  MTPS=106427             ;INSTR EQUATE THAT MOVES BYTE TO PSW
```

```
                 177522          TMAIN= 177522          ;MAINTENCE REGISTER(FOR USE WITH KDF11-B)


                 000000          AMSGTY=0
                 000000          AFATAL=0
                 000000          ATESTN=0
                 000000          APASS=0
                 000000          ADEVCT=0
                 000000          AUNIT=0
                 000000          AMSGAD=0
                 000000          AMSGLG=0
                 000000          AENV=0
                 000000          AENVM=0
                 000000          ASWREG=0
                 000000          AUSWR=0
                 000000          ACPUOP=0
                 000000          AMAMS1=0
                 000000          AMTYP1=0
                 000000          AMADR1=0
                 000000          AMAMS2=0
                 000000          AMTYP2=0
                 000000          AMADR2=0
                 000000          AMAMS3=0
                 000000          AMTYP3=0
                 000000          AMADR3=0
                 000000          AMAMS4=0
                 000000          AMTYP4=0
                 000000          AMADR4=0
                 000000          AVECT2=0
                 000000          ACDW1=0
                 000000          ACDW2=0
                 000000          ADDW0=0
                 000000          ADDW1=0
                 000000          ADDW2=0
                 000000          ADDW3=0
                 000000          ADDW4=0
                 000000          ADDW5=0
                 000000          ADDW6=0
                 000000          ADDW7=0
                 000000          ADDW8=0
                 000000          ADDW9=0
                 000000          ADDW10=0
                 000000          ADDW11=0
                 000000          ADDW12=0
                 000000          ADDW13=0
                 000000          ADDW14=0
                 000000          ADDW15=0


                                 .SBTTL  TRAP CATCHER

                 000000          .ASECT
000000   014052                 $TRAP
000002   000340                 340
                 000000                  .=0
                                 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
```

```
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
000174  000000         DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000         SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001550          JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
                        000100          .=100
000100  000104  000200  000002          .WORD   104,200,2       ;IF 'B EVENT' ON Q BUS IS CONNECTED
                                                                ;IGNORE IT'S INTERRUPT - JUST DO A RTI
                                .SBTTL  ACT11 HOOKS

                                ;;**********************************************************************
                                ;HOOKS REQUIRED BY ACT11
                        000106          $SVPC=.                 ;SAVE PC
                        000046          .=46
000046  010042                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052          .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                        000106          .=$SVPC                 ;; RESTORE PC
                        001000          .=1000
                                .SBTTL  APT PARAMETER BLOCK

                                ;;**********************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;**********************************************************************
                        001000          .$X=.   ;;SAVE CURRENT LOCATION
                        000024          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
                        000044          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
                        001000          .=.$X   ;;RESET LOCATION COUNTER
                                ;;**********************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001174                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000031                  $TSTM:  .WORD   25.     ;;RUN TIM OF LONGEST TEST
001006  000006                  $PASTM: .WORD   6.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000144                  $UNITM: .WORD   100.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
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
```

```
6300  001110  000000                  . PERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
6301  001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
6302  001114     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
6303  001115     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
6304  001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
6305  001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
6306  001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
6307  001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
6308  001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
6309  001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
6310  001132  000000                          .WORD   0
6311  001134     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
6312  001135     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
6313  001136  000000                          .WORD   0
6314  001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
6315  001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
6316  001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
6317  001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
6318  001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
6319  001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
6320  001154     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
6321  001155     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
6322  001156     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
6323  001157     000                  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
6324  001160  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
6325  001162  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
6326  001164  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
6327  001170     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
6328  001171     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
6329  001172  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
6330                                  ;;**************************************************************************
6331                                  .SBTTL  APT MAILBOX-ETABLE
6332
6333                                  ;;**************************************************************************
6334                                  .EVEN
6335  001174                          $MAIL:                          ;;APT MAILBOX
6336  001174  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
6337  001176  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
6338  001200  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
6339  001202  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
6340  001204  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
6341  001206  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
6342  001210  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
6343  001212  000000                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
6344  001214                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
6345  001214     000                  $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
6346  001215     000                  $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
6347  001216  000000                  $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
6348  001220  000000                  $USWR:  .WORD   AUSWR           ;;USER SWITCHES
6349  001222  000000                  $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
6350                                  ;*                              BITS 15-11=CPU TYPE
6351                                  ;*                                   11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
6352                                  ;*                                   11/70=06,PDQ=07,Q=10
6353                                  ;*                              BIT 10=REAL TIME CLOCK
6354                                  ;*                              BIT  9=FLOATING POINT PROCESSOR
6355                                  ;*                              BIT  8=MEMORY MANAGEMENT
```

```
 6356  001224  000
 6357  001225  000
 6358
 6359
 6360
 6361
 6362  001226  000000
 6363
 6364  001230  000
 6365  001231  000
 6366  001232  000000
 6367  001234  000
 6368  001235  000
 6369  001236  000000
 6370  001240  000
 6371  001241  000
 6372  001242  000000
 6373  001244  000124
 6374  001246  000000
 6375  001250  172410
 6376  001252  000001
 6377  001254  000000
 6378  001256  000000
 6379  001260  000000
 6380  001262  000000
 6381  001264  000000
 6382  001266  000000
 6383  001270  000000
 6384  001272  000000
 6385  001274  000000
 6386  001276  000000
 6387  001300  000000
 6388  001302  000000
 6389  001304  000000
 6390  001306  000000
 6391  001310  000000
 6392  001312  000000
 6393  001314  000000
 6394  001316  000000
 6395
 6396
 6397  001320
 6398
 6399
 6400
 6401
 6402
 6403
 6404
 6405
 6406
 6407
 6408
 6409
 6410
 6411

$MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
$MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
;*                              900 NSEC CORE=001
;*                              300 NSEC BIPOLAR=002
;*                              500 NSEC MOS=003
$MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
$MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
$MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
$MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
$MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
$MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
$MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
$MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
$MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
$VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
$VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
$BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
$DEVM:  .WORD   ADEVM   ;;DEVICE MAP
$CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
$CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
$DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
$DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
$DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
$DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
$DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
$DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
$DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
$DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
$DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
$DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
$DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
$DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
$DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
$DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
$DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
$DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15


$ETEND:

.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
```

```
 6412
 6413  001320                          $ERRTB:
 6414                                  ;ERROR  1
 6415  001320  014134                          EM1             ;REG TIMEOUT ER
 6416  001322  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6417  001324  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6418  001326  000000                          0
 6419
 6420                                  ;ERROR  2
 6421  001330  014153                          EM2             ;REG READ/WRITE ER
 6422  001332  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6423  001334  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6424  001336  000000                          0
 6425
 6426                                  ;ERROR  3
 6427  001340  014175                          EM3             ;BUS RESET ER
 6428  001342  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6429  001344  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6430  001346  000000                          0
 6431
 6432                                  ;ERROR  4
 6433  001350  014212                          EM4             ;FNCT BITS FAILED TO SET STAT BITS
 6434  001352  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6435  001354  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6436  001356  000000                          0
 6437
 6438                                  ;ERROR  5
 6439  001360  014254                          EM5             ;READY INTR FAILURE
 6440  001362  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6441  001364  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6442  001366  000000                          0
 6443
 6444                                  ;ERROR  6
 6445  001370  014277                          EM6             ;READY CLR OR SET ER
 6446  001372  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6447  001374  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6448  001376  000000                          0
 6449                                  ;ERROR  7
 6450  001400  014323                          EM7             ;STATUS ER ON XFER
 6451  001402  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6452  001404  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6453  001406  000000                          0
 6454
 6455                                  ;ERROR  10
 6456  001410  014345                          EM10            ;WORD COUNT ER ON XFER
 6457  001412  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6458  001414  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6459  001416  000000                          0
 6460
 6461                                  ;ERROR  11
 6462  001420  014373                          EM11            ;BUFFER ADRS ER ON XFER
 6463  001422  015025                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6464  001424  015176                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6465  001426  000000                          0
 6466
 6467                                  ;ERROR  12
```

```
 6468  001430  014422                    EM12            ;DATA ER FROM MEM
 6469  001432  015072                    DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
 6470  001434  015212                    DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
 6471  001436  000000                    0
 6472
 6473                          ;ERROR  13
 6474  001440  014443                    EM13            ;DATA ER TO MEM
 6475  001442  015072                    DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
 6476  001444  015212                    DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
 6477  001446  000000                    0
 6478
 6479                          ;ERROR  14
 6480  001450  014462                    EM14            ;SINGLE CYCLE OFF DID NOT LOCK OUT CPU
 6481  001452  015147                    DH3             ;ERRPC  TSTNUM  BUSADR
 6482  001454  015230                    DT3             ;$ERRPC TSTNUM  $BDADR
 6483  001456  000000                    0
 6484
 6485                          ;ERROR  15
 6486  001460  014530                    EM15            ;SINGLE CYCLE ON LOCKED OUT CPU
 6487  001462  015147                    DH3             ;ERRPC  TSTNUM  BUSADR
 6488  001464  015230                    DT3             ;$ERRPC TSTNUM  $BDADR
 6489  001466  000000                    0
 6490
 6491                          ;ERROR  16
 6492  001470  014716                    EM16            ;NEX LOGIC ER
 6493  001472  015025                    DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
 6494  001474  015176                    DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6495  001476  000000                    0
 6496
 6497                          ;ERROR  17
 6498  001500  014733                    EM17            ;CYCLE FAILED TO CLK DBR (IN)
 6499  001502  015025                    DH1             ;ERRPC  TSTNUM  BDADR   GDDAT   BDDAT
 6500  001504  015176                    DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
 6501  001506  000000                    0
 6502                          ;ERROR  20
 6503  001510  014770                    EM20            ;DATA ER FROM I/O PAGE (XCSR)
 6504  001512  015072                    DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
 6505  001514  015212                    DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
 6506  001516  000000                    0
 6507
 6508
 6509                          ;DRV11B BUS REGISTER ADDRESS POINTERS
 6510
 6511  001520  172410          DRVWCR: 172410          ;WORD COUNT
 6512  001522  172412          DRVBAR: 172412          ;BUFFER ADDRESS
 6513  001524  172414          DRVCSR: 172414          ;COMMAND/STATUS
 6514  001526  172416          DRVDBR: 172416          ;DATA BUFFER
 6515
 6516                          ;DRV11B VECTOR ADDRESS POINTERS
 6517
 6518  001530  000124          DRVCT0: 124             ;READY, NEX & INCOMPLETE DATIO VECTOR
 6519  001532  000126          DRVCT2: 126             ;NEW PSW ON INTR
 6520
 6521                          ;COMMON PROGRAM LOCATION(S)
 6522
 6523  001534  000000          TSTNUM: 0               ;CONTAINS TEST NUMBER ON ERROR
```

```
                                        DMAP:   1               ;DEVICE MAP - EA BIT SAYS TEST THAT DRV11B
                                        CORSZ:  0               ;CONTAINS 1ST NON-EXISTANT MEM ADRS
                                        DBUFP:  DBUF            ;CONTAINS CURRENT 4K NPR BUFFER ADRS
                                        IOPAGE: 0               ;CONTAINS ADDRESS FOR I/O TRANSFERS
                                        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  012432  000020                  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001576  012737  000340  000022                  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001604  012737  012070  000030                  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001612  012737  000340  000032                  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001620  012737  014052  000034                  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001626  012737  000340  000036                  MOV     #340,@#TRAPVEC+2;LEVEL 7
001634  012737  013646  000024                  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001642  012737  000340  000026                  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
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
```

```
                                        MOV     $VECT1,R1       ;GET BASE VECTOR ADRS
                                        BIC     #170000,R1      ;CLR OUT PRIORITY BITS
                                SETUP3: MOV     R1,(R0)+        ;
                                        ADD     #2,R1           ;POINT TO NEXT
                                        CMP     #DRVCT2+2,R0    ;ALL DONE?
                                        BNE     SETUP3          ;BR IF NOT
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
                                        INC     #-1             ;;FIRST TIME?
                                        BNE     64$             ;;BRANCH IF NO
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
                                        TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
                                        BNE     66$             ;;BRANCH IF YES
                                        CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
                                        BEQ     66$             ;;BRANCH IF YES
                                        CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
                                        BNE     67$             ;;BRANCH IF NO
                                        GTSWR                   ;;GET SOFT-SWR SETTINGS
                                        BR      67$
                                66$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
                                67$:
                                        BR      64$             ;;GET OVER THE ASCIZ
                                65$:    .ASCIZ  <CRLF>@MD-11-CVDRA-C  DRV11B DMA INTERFACE DIAG  @<CRLF>
                                        .EVEN
                                64$:
                                ;       INC     #-1             ;FIRST PASS?                    ;DD001 INSTRUCTIONS
                                ;       BNE     100$            ;BRANCH IF NO                   ;DD001 NO LONGER
                                ;       CMPB    #1,$AUTOB       ;MANUAL INTERVENTION PERMITTED? ;DD001 NECESSARY
                                ;       BEQ     100$            ;BR IF NO                       ;DD001
                                        MOV     @#10,-(SP)      ;SAVE ILLEGAL INSTRUCTION       ;DD001
                                        MOV     @#12,-(SP)      ;TRAP VECTOR                    ;DD001
                                        MOV     #100$,@#10      ;                               ;DD001
                                        MFPT                    ;IF CPU TYPE IS 11/23           ;DD001
                                        BISB    #BIT07,@SWR     ;THEN SET BIT 7 IN SWREG        ;DD001
                                100$:   MOV     (SP)+,@#12      ;AND RESTORE TRAP VECTOR        ;DD001
                                        MOV     (SP)+,@#10      ;                               ;DD001
                                        TST     CORSZ           ;TEST IF FIRST PASS
                                        BNE     CORSZR          ;BR IF NOT
                                        TYPE    ,WARN           ;TELL THE OPERATOR TO TURN OFF DMA REFRESH
                                ;;*****************************************************************
                                ;LET'S SEE HOW MUCH MEM WE HAVE
                                ;;*****************************************************************
                                CORSZR: MOV     #20000,R0       ;USE R0 TO LOOK
                                        MOV     #2$,@#ERRVEC    ;SET UP TIME OUT RETURN ADRS
                                1$:     TST     (R0)+           ;TAKE A LOOK
                                        BR      1$              ;UNTIL TIMEOUT
                                2$:     BIC     #17777,R0       ;POINT TO 1ST NON-EXSISTANT 4K BLK
                                        MOV     R0,CORSZ        ;SAVE FOR LATER
```

```
 6629  002316  012737  000006  000004          MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE VECTOR
 6630  002324  012737  015364  001542          MOV     #DBUF,DBUFP         ;INITIALIZE TO LOWEST 4K
 6631  002332  012737  000000  001206          MOV     #0,$UNIT            ;SET UP UNIT COUNT
 6632  002340  013737  001252  001536          MOV     $DEVM,DMAP          ;GET THE # & POSITION OF DRV11B'S
 6633  002346  042737  177400  001536          BIC     #177400,DMAP        ;UP TO 8 ONLY
 6634  002354  001406                          BEQ     RESTRT              ;GO CONTINUE AS IF SOMETHING WAS SELECTED
 6635  002356  032737  000001  001536          BIT     #1,DMAP             ;IS 1ST DRV11B SELECTED?
 6636  002364  001002                          BNE     RESTRT              ;BR IF SO
 6637  002366  000137  007660                  JMP     NXDEV1              ;NO - GO ADVANCE BASE DRV11B ADDRESSES
 6638  002372  106427  000200          RESTRT: MTPS    #200                ;SET PRIORITY TO HIGHEST LEVEL
 6639  002376  012706  001100                  MOV     #STACK,SP           ;ALWAYS RESET STACK PTR
 6640  002402  013737  001206  001204          MOV     $UNIT,$DEVCT        ;LOAD APT COUNTER
 6641  002410  013700  001206                  MOV     $UNIT,R0            ;MAKE AN INDEX
 6642  002414  006300                          ASL     R0                  ; VALUE
 6643  002416  013760  001520  001260          MOV     DRVWCR,$DDW0(R0)            ;SAVE THE BUS ADDRESS
 6644  002424  000005                          RESET                       ;INITIALIZE DRV11B BEFORE TESTING
 6645                                  ;;**************************************************************************
 6646                                  ;*TEST 1         TEST THAT ALL DRV11B REGS ARE ACCESSIBLE
 6647                                  ;;**************************************************************************
 6648  002426  000240                  TST1:   <NOP>
 6649  002430  012737  002444  001106          MOV     #10$,$LPADR         ;;SET SCOPE LOOP ADDRESS
 6650  002436  012737  000001  001200          MOV     #1,$TESTN           ;;SET TEST NUMBER IN APT MAIL BOX
 6651  002444  112737  000001  001102  10$:     MOVB    #1,$TSTNM           ;SET TO TEST #1
 6652  002452  012737  002506  001110          MOV     #1$,$LPERR          ;SET UP SCOPE LOOP ADRS
 6653  002460  005037  001124                  CLR     $GDDAT              ;NO DATA COMPARE
 6654  002464  005037  001126                  CLR     $BDDAT              ;NO DATA COMPARE
 6655  002470  012737  002524  000004          MOV     #2$,@#ERRVEC        ;SET UP TIMEOUT RETURN ADDRS
 6656  002476  013700  001520                  MOV     DRVWCR,R0           ;SET UP 1ST DRV11 BUS ADRS
 6657  002502  012701  000004                  MOV     #4,R1               ;SET UP REG COUNT
 6658  002506  010037  001122          1$:     MOV     R0,$BDADR           ;SET UP CURRENT DRV BUS ADRS
 6659  002512  005710                          TST     (R0)                ;SEE IF THERE
 6660  002514  005720                          TST     (R0)+               ;BUMP TO NEXT
 6661  002516  005301                          DEC     R1                  ;COUNT 4 OF THEM
 6662  002520  001403                          BEQ     3$                  ;BR IF ALL DONE
 6663  002522  000771                          BR      1$                  ;TRY NEXT
 6664  002524  022626                  2$:     CMP     (SP)+,(SP)+         ;FIX STACK SINCE NO RTI
 6665  002526  104001                          ERROR   1                   ;BUS ADRS INDICATED DID NOT RESPOND
 6666  002530  012737  000006  000004  3$:     MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE LOC 4
 6667
 6668                                  ;;**************************************************************************
 6669                                  ;*TEST 2         TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
 6670                                  ;;**************************************************************************
 6671  002536  000004                  TST2:   SCOPE
 6672  002540  012737  002564  001110          MOV     #1$,$LPERR          ;SET UP SCOPE LOOP ADRS
 6673  002546  013737  001520  001122          MOV     DRVWCR,$BDADR       ;SET UP WC REG ADRS
 6674  002554  005000                          CLR     R0                  ;R0 SAYS SHIFT PTRN WHEN 0
 6675  002556  012737  177776  001124          MOV     #-2,$GDDAT          ;FLOAT 0 RIGHT TO LEFT
 6676  002564  013777  001124  176726  1$:     MOV     $GDDAT,@DRVWCR      ;LD WC
 6677  002572  017737  176722  001126          MOV     @DRVWCR,$BDDAT      ;READ IT BACK
 6678  002600  023737  001124  001126          CMP     $GDDAT,$BDDAT       ;CORRECT?
 6679  002606  001401                          BEQ     2$                  ;BR IF SO
 6680  002610  104002                          ERROR   2                   ;WORD COUNT WRITE/READ FAILURE
 6681  002612  005137  001124          2$:     COM     $GDDAT              ;COMPELEMENT ZERO
 6682  002616  005100                          COM     R0                  ;R0 SAYS SHIFT LEFT WHEN = 0
 6683  002620  001361                          BNE     1$                  ;TRY THE COMPLEMENT IF R0 NOT 0
 6684  002622  006337  001124                  ASL     $GDDAT              ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
```

```
 6685  002626  005237  001124                 INC     $GDDAT          ;KEEP LSB SET
 6686  002632  103754                         BCS     1$              ;AGAIN TILL ALL PATRNS DONE
 6687
 6688                                 ;;*******************************************************************
 6689                                 ;*TEST 3        TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
 6690                                 ;;*******************************************************************
 6691  002634  000004                 TST3:   SCOPE
 6692  002636  012737  002662  001110         MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6693  002644  013737  001522  001122         MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
 6694  002652  005000                         CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
 6695  002654  012737  177774  001124         MOV     #-4,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
 6696  002662  013777  001124  176632 1$:     MOV     $GDDAT,@DRVBAR  ;LD BA
 6697  002670  017737  176626  001126         MOV     @DRVBAR,$BDDAT  ;READ IT BACK
 6698  002676  042737  000001  001126         BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
 6699  002704  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;CORRECT?
 6700  002712  001401                         BEQ     2$              ;BR IF SO
 6701  002714  104002                         ERROR   2               ;BUS ADRS WRITE/READ FAILURE
 6702  002716  005137  001124         2$:     COM     $GDDAT          ;COMPLEMENT ZERO
 6703  002722  042737  000001  001124         BIC     #BIT00,$GDDAT   ;BIT 00 NOT INVOLVED
 6704  002730  005100                         COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
 6705  002732  001353                         BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
 6706  002734  006337  001124                 ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
 6707  002740  103004                         BCC     TST4            ;;NEXT TEST IF BIT 15 DONE
 6708  002742  062737  000002  001124         ADD     #2,$GDDAT       ;KEEP ADDR LSB SET
 6709  0C2750  000744                         BR      1$              ;AGAIN TILL ALL PATTERNS DONE
 6710
 6711                                 ;;*******************************************************************
 6712                                 ;*TEST 4        TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
 6713                                 ;;*******************************************************************
 6714  002752  000004                 TST4:   SCOPE
 6715  002754  012737  003000  001110         MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6716  002762  013737  001526  001122         MOV     DRVDBR,$BDADR   ;SET UP DB REG ADRS
 6717  002770  005000                         CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
 6718  002772  012737  177776  001124         MOV     #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
 6719  003000  013777  001124  176520 1$:     MOV     $GDDAT,@DRVDBR  ;LD DB
 6720  003006  017737  176514  001126         MOV     @DRVDBR,$BDDAT  ;READ IT BACK
 6721  003014  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;CORRECT?
 6722  003022  001401                         BEQ     2$              ;BR IF SO
 6723  003024  104002                         ERROR   2               ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
 6724  003026  005137  001124         2$:     COM     $GDDAT          ;COMPLEMENT ZERO
 6725  003032  005100                         COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
 6726  003034  001361                         BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
 6727  003036  006337  001124                 ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
 6728  003042  005237  001124                 INC     $GDDAT          ;KEEP LSB SET
 6729  003046  103754                         BCS     1$              ;AGAIN TILL ALL PATRNS DONE
 6730
 6731                                 ;;*******************************************************************
 6732                                 ;*TEST 5        TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE
 6733                                 ;;*******************************************************************
 6734  003050  000004                 TST5:   SCOPE
 6735  003052  013700  001526                 MOV     DRVDBR,R0       ;GET DB REG ADRS
 6736  003056  010037  001122                 MOV     R0,$BDADR       ;SET UP DB REG ADRS
 6737  003062  005010                         CLR     (R0)            ;ZERO DATA BUFFER REG
 6738  003064  012737  177177  001124         MOV     #177177,$GDDAT  ;LD EXPECTED
 6739  003072  012737  077776  001126         MOV     #77776,$BDDAT   ;SEND DATA FROM "BDDAT"
 6740  003100  153760  001126  000001         BISB    $BDDAT,1(R0)    ;LOAD HI BYTE DB
```

```
6741  003106  153710  001127                  BISB    $BDDAT+1,(R0)   ;LOAD LO BYTE DB
6742  003112  011037  001126                  MOV     (R0),$BDDAT     ;READ IT BACK
6743  003116  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6744  003124  001401                          BEQ     TST6            ;;NEXT TEST IF SO
6745  003126  104002                          ERROR   2               ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG
6746
6747                                  ;;**********************************************************************
6748                                  ;*TEST 6        TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
6749                                  ;;**********************************************************************
6750  003130  000004                  TST6:   SCOPE
6751  003132  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
6752  003140  005037  001124                  CLR     $GDDAT          ;LD EXPECTED
6753  003144  012777  177777  176346          MOV     #-1,@DRVWCR     ;SET ALL BITS - WC REG
6754  003152  012777  177776  176342          MOV     #-2,@DRVBAR     ;SET ALL BITS - BUS ADRS REG
6755  003160  012777  177777  176340          MOV     #-1,@DRVDBR     ;SET ALL BITS - DB OUT REG
6756  003166  000005                          RESET                   ;DO A BUS RESET
6757  003170  017737  176324  001126          MOV     @DRVWCR,$BDDAT  ;READ WC REG
6758  003176  001404                          BEQ     1$              ;BR IF CLRED
6759  003200  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WC REG ADRS
6760  003206  104003                          ERROR   3               ;RESET FAILED TO CLR WC REG
6761  003210  017737  176306  001126  1$:     MOV     @DRVBAR,$BDDAT  ;READ BUS ADRS REG
6762  003216  042737  000001  001126          BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
6763  003224  001404                          BEQ     2$              ;BR IF CLRED
6764  003226  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
6765  003234  104003                          ERROR   3               ;RESET FAILED TO CLR BUS ADRS REG
6766  003236  017737  176264  001126  2$:     MOV     @DRVDBR,$BDDAT  ;READ DATA BUFFER REG
6767  003244  001404                          BEQ     TST7            ;;NEXT TEST IF CLRED
6768  003246  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DB REG ADRS
6769  003254  104003                          ERROR   3               ;RESET FAILED TO CLR DATA BUFFER OUT REG
6770
6771                                  ;;**********************************************************************
6772                                  ;*TEST 7        TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
6773                                  ;;**********************************************************************
6774  003256  000004                  TST7:   SCOPE
6775  003260  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
6776  003266  106427  000200                  MTPS    #200            ;DON'T WANT ANY INTRS
6777  003272  004537  010076                  JSR     R5,SETVEC       ;SET UP INTR RETURN ADRS IN CASE
6778  003276  003374                          3$                      ;RETURN TO 3$ ON ILLEGAL INTR
6779  003300  013737  003320  001110          MOV     1$,$LPERR       ;SET UP SCOPE LOOP ADRS
6780  003306  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
6781  003314  012700  160000                  MOV     #160000,R0      ;START AT 0 - HI BITS FOR NOISE
6782  003320  010037  001124          1$:     MOV     R0,$GDDAT       ;LD EXPECTED
6783  003324  042737  167201  001124          BIC     #167201,$GDDAT  ;MASK TO WRITEABLE BITS
6784  003332  010077  176166                  MOV     R0,@DRVCSR      ;LD CSR
6785  003336  017737  176162  001126          MOV     @DRVCSR,$BDDAT  ;READ IT BACK
6786  003344  042737  007200  001126          BIC     #7200,$BDDAT    ;DON'T LOOK AT STAT & RDY BITS
6787  003352  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6788  003360  001401                          BEQ     2$              ;BR IF SO
6789  003362  104002                          ERROR   2               ;CONTROL/STATUS REG WRITE/READ FAILURE
6790  003364  062700  000002          2$:     ADD     #2,R0           ;ADVANCE COUNT PATTERN
6791  003370  001353                          BNE     1$              ;WRITE NEXT PATTERN IF NOT ALL TESTED
6792  003372  000413                          BR      4$              ;GO RESTORE VECTOR
6793  003374  022626                  3$:     CMP     (SP)+,(SP)+     ;FIX STACK - SHOULD NOT HAVE INTR'ED
6794  003376  052737  000200  001124          BIS     #200,$GDDAT     ;CORRECT EXPECTED
6795  003404  017737  176114  001126          MOV     @DRVCSR,$BDDAT  ;READ CSR
6796  003412  042737  007000  001126          BIC     #7000,$BDDAT    ;DON'T WANT 'STAT' BITS
```

```
                        ERROR   5               ;CPU FAILED TO LOCK OUT DRV11B INTR REQ
4$:                     JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                ;;**************************************************************************
                ;*TEST 10        TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
                ;;**************************************************************************
TST10:          SCOPE
                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                        MOV     #200,$GDDAT     ;LD EXPECTED
                        MOV     #-2,@DRVCSR     ;LD ALL CSR BITS
                        RESET                   ;DO A BUS RESET
                        MOV     @DRVCSR,$BDDAT  ;READ CSR
                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                        BEQ     TST11           ;;NEXT TEST IF CSR CLRED & READY SET
                        ERROR   3               ;RESET FAILED TO SET UP THE CSR

                ;;**************************************************************************
                ;*TEST 11        TEST THAT THE CSR IS BYTE ADDRESSABLE
                ;;**************************************************************************
TST11:          SCOPE
                        MOV     DRVCSR,R0       ;GET CSR ADRS
                        MOV     R0,$BDADR       ;SET UP CSR ADRS
                        CLR     (R0)            ;ZERO CSR
                        MOV     #10300,$GDDAT   ;LD EXPECTED
                        MOV     #40020,$BDDAT   ;SEND DATA FROM "BDDAT" - USE MAIN + IE
                        BISB    $BDDAT,1(R0)    ;LOAD HI BYTE CSR
                        BISB    $BDDAT+1,(R0)   ;LOAD LO BYTE CSR
                        MOV     (R0),$BDDAT     ;READ IT BACK
                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                        BEQ     1$              ;BR IF SO
                        ERROR   2               ;DATA ERROR ON BYTE ADDRESSING THE CSR
1$:                     CLR     (R0)            ;ZERO CSR BEFORE ADVANCING

                ;;**************************************************************************
                ;*TEST 12        TEST THAT THE 3 "FNCT" BITS CONTROL THE 3 "STAT" BITS (COUNT PTRN)
                ;;**************************************************************************
TST12:          SCOPE
                        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                        MOV     #7000,$GDDAT    ;LD EXPECTED
                        MOV     #16,R0          ;R0 CONTAINS "FNCT" BITS WRITTEN
1$:                     MOV     R0,@DRVCSR      ;WRITE INTO "FNCT" BITS
                        MOV     @DRVCSR,$BDDAT  ;READ BACK THRU "STAT" BITS
                        BIC     #170777,$BDDAT  ;MASK TO "STAT" BITS ONLY
                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                        BEQ     2$              ;BR IF SO
                        ERROR   4               ;"FNCT" BITS FAILED TO SET "STAT" BITS (LOOP BACK)
2$:                     SUB     #1000,$GDDAT    ;CHANGE TO NEXT EXPECTED
                        SUB     #2,R0           ;DECREASE COUNT PATTERN
                        BPL     1$              ;DO AGAIN UNTIL 0 TESTED

                ;;**************************************************************************
                ;*TEST 13        TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0
                ;;**************************************************************************
TST13:          SCOPE
```

```
6853  003664  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
6854  003672  106427  000200                  MTPS    #200            ;DONT WANT INTR YET
6855  003676  000005                          RESET                   ;SET THE READY FLAG BY INIT
6856  003700  012777  003760  175622          MOV     #1$,@DRVCTO     ;SET UP PREMATURE INTR RETURN ADRS
6857  003706  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
6858  003714  012737  000300  001124          MOV     #300,$GDDAT     ;LD EXPECTED (READY + IE)
6859  003722  106427  000000                  MTPS    #0              ;ALLOW AN INTR
6860  003726  021616                          CMP     (SP),(SP)       ;STALL
6861  003730  012777  003774  175572          MOV     #2$,@DRVCTO     ;SET UP EXPECTED INTR RETURN ADRS
6862  003736  052777  000100  175560          BIS     #BIT6,@DRVCSR   ;ENABLE THE EXPECTED INTERRUPT
6863  003744  021616                          CMP     (SP),(SP)       ;STALL
6864  003746  017737  175552  001126          MOV     @DRVCSR,$BDDAT  ;GET THE CSR
6865  003754  104005                          ERROR   5               ;READY FAILED TO CAUSE AN INTERRUPT
6866  003756  000417                          BR      3$              ;GO RESTORE VECTOR
6867  003760  022626                  1$:     CMP     (SP)+,(SP)+     ;SHOULD NEVER GET HERE - IE NOT WORKING?
6868  003762  017737  175536  001126          MOV     @DRVCSR,$BDDAT  ;GET THE CSR
6869  003770  104005                          ERROR   5               ;READY INTERRUPTED WITHOUT THE IE BIT
6870  003772  000411                          BR      3$              ;GO RESTORE VECTOR
6871  003774  022626                  2$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RETURN
6872  003776  017737  175522  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
6873  004004  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6874  004012  001401                          BEQ     3$              ;BR IF SO
6875  004014  104005                          ERROR   5               ;INCORRECT STATUS ON READY INTR
6876  004016  004737  010116          3$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
6877
6878                                  ;;**********************************************************************
6879                                  ;*TEST 14        TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
6880                                  ;;**********************************************************************
6881  004022  000004                  TST14:  SCOPE
6882  004024  106427  000200                  MTPS    #200            ;DONT WANT ANY INTRS
6883  004030  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
6884  004036  005037  001124                  CLR     $GDDAT          ;EXPECT 0
6885  004042  012777  000001  175454          MOV     #1,@DRVCSR      ;SET GO WHICH SHOULD CLR READY
6886  004050  017737  175450  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
6887  004056  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6888  004064  001401                          BEQ     1$              ;BR IF SO
6889  004066  104006                          ERROR   6               ;THE GO BIT FAILED TO CLR READY
6890  004070  052777  000004  175426  1$:     BIS     #4,@DRVCSR      ;FNCT 2 SHOULD SET READY
6891  004076  012737  002204  001124          MOV     #2204,$GDDAT    ;LD EXPECTED
6892  004104  017737  175414  001126          MOV     @DRVCSR,$BDDAT  ;GET CSR
6893  004112  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6894  004120  001401                          BEQ     TST15           ;;NEXT TEST IF SET
6895  004122  104006                          ERROR   6               ;FNCT 2 (VIA ATTN) FAILED TO SET READY
6896
6897                                  ;;**********************************************************************
6898                                  ;*TEST 15        TEST THAT READY CONTROLS 'BAR' BIT00
6899                                  ;;**********************************************************************
6900  004124  000004                  TST15:  SCOPE
6901  004126  012777  000004  175370          MOV     #4,@DRVCSR      ;SET READY
6902  004134  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
6903  004142  012737  000001  001124          MOV     #1,$GDDAT       ;EXPECT LSB OF BAR
6904  004150  005077  175346                  CLR     @DRVBAR         ;CLR BAR
6905  004154  017737  175342  001126          MOV     @DRVBAR,$BDDAT  ;READ BAR
6906  004162  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
6907  004170  001401                          BEQ     1$              ;BR IF SO
6908  004172  104002                          ERROR   2               ;A00 FAILED TO READ A ONE(SB TIED TO RDY)
```

```
 6909  004174  012777  000001  175322  1$:     MOV     #1,@DRVCSR      ;SET GO(CLRS BAR BIT00)
 6910  004202  017737  175314  001126          MOV     @DRVBAR,$BDDAT  ;READ BAR
 6911  004210  001403                          BEQ     2$              ;BR IF ZERO
 6912  004212  005037  001124                  CLR     $GDDAT          ;EXPECTED ZERO
 6913  004216  104002                          ERROR   2               ;WHEN RDY CLRED-A00 FAILED TO READ A ZERO
 6914  004220  012777  000004  175276  2$:     MOV     #4,@DRVCSR      ;INSURE RDY SET BEFORE ADVANCING
 6915
 6916                                  ;;*****************************************************************
 6917                                  ;*TEST 16        TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
 6918                                  ;;*****************************************************************
 6919  004226  000004                  TST16:  SCOPE
 6920  004230  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 6921  004236  012737  125252  001124          MOV     #125252,$GDDAT  ;LD EXPECTED
 6922  004244  013777  001124  175254          MOV     $GDDAT,@DRVDBR  ;LD DBR WITH #125252
 6923  004252  012777  000400  175244          MOV     #400,@DRVCSR    ;SET CYCLE - SHOULD CLK DBR (IN)
 6924  004260  012777  052525  175240          MOV     #52525,@DRVDBR  ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
 6925  004266  017737  175234  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
 6926  004274  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6927  004302  001401                          BEQ     1$              ;BR IF SO
 6928  004304  104017                          ERROR   17              ;CYCLE DID NOT LATCH DBR (IN) DATA
 6929  004306  042777  000400  175210  1$:     BIC     #400,@DRVCSR    ;REMOVE CYCLE
 6930  004314  012737  052525  001124          MOV     #52525,$GDDAT   ;NOW EXPECT #52525
 6931  004322  017737  175200  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
 6932  004330  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 6933  004336  001401                          BEQ     TST17           ;;NEXT TEST IF SO
 6934  004340  104002                          ERROR   2               ;DBR FAILED TO READ WHEN CYCLE CLRED (NORMAL)
 6935
 6936                                  ;;*****************************************************************
 6937                                  ;*TEST 17        TEST SINGLE "DATI" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
 6938                                  ;;*****************************************************************
 6939  004342  000004                  TST17:  SCOPE
 6940  004344  012737  004366  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6941  004352  004537  010076                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
 6942  004356  004462                          2$                      ;RETURN TO 2$ ON INTR
 6943  004360  012700  177776                  MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
 6944  004364  005001                          CLR     R1              ;R1 CONTROLS DATA SHIFTING
 6945  004366  012777  177777  175124  1$:     MOV     #-1,@DRVWCR     ;DO ONE XFER
 6946  004374  012777  015364  175120          MOV     #DBUF,@DRVBAR   ;GET DATA WORD FROM "DBUF"
 6947  004402  010037  015364                  MOV     R0,DBUF         ;SET UP MEM DATA
 6948  004406  012777  000101  175110          MOV     #101,@DRVCSR    ;SET IE & GO
 6949  004414  052777  000400  175102          BIS     #400,@DRVCSR    ;SET CYCLE
 6950  004422  106427  000000                  MTPS    #0              ;ENABLE THE INTR
 6951  004426  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6952  004434  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
 6953  004442  017737  175056  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 6954  004450  042777  000100  175046          BIC     #100,@DRVCSR    ;CLR IE
 6955  004456  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
 6956  004460  000442                          BR      4$              ;GO RESTORE VECTOR
 6957  004462  022626                  2$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
 6958  004464  004537  010142                  JSR     R5,CKSTAT       ;GO CHECK STATUS
 6959  004470  000700                          700                     ;CSR STATUS EXPECTED
 6960  004472  000001                          1                       ;# OF XFERS
 6961  004474  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
 6962  004476  000433                          BR      4$              ;GO RESTORE VECTOR
 6963  004500  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
 6964  004502  000431                          BR      4$              ;GO RESTORE VECTOR
```

```
 6965  004504  104011                        ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
 6966  004506  000427                        BR      4$              ;GO RESTORE VECTOR
 6967  004510  012737  015364  001120        MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
 6968  004516  010037  001124                MOV     R0,$GDDAT       ;LD EXPECTED
 6969  004522  017737  175000  001126        MOV     @DRVDBR,$BDDAT  ;READ DATA XFERED
 6970  004530  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
 6971  004536  001405                        BEQ     3$              ;BR IF SO
 6972  004540  013737  001526  001122        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 6973  004546  104012                        ERROR   12              ;DATA ER - DBR CONTAINS WRONG DATA
 6974  004550  000406                        BR      4$              ;GO RESTORE VECTOR
 6975  004552  005100                3$:     COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
 6976  004554  005101                        COM     R1              ;KEEP TRACK OF COMPLEMENT
 6977  004556  001303                        BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
 6978  004560  006300                        ASL     R0              ;WAS DONE - NOW SHIFT ZERO LEFT
 6979  004562  005200                        INC     R0              ;KEEP LSB SET
 6980  004564  103700                        BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
 6981  004566  004737  010116        4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
 6982
 6983                                ;;**************************************************************************
 6984                                ;*TEST 20        TEST SINGLE "DATO" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
 6985                                ;;**************************************************************************
 6986  004572  000004                TST20:  SCOPE
 6987  004574  012737  004616  001110        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 6988  004602  004537  010076                JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
 6989  004606  004704                        2$                      ;RETURN TO 2$ ON INTR
 6990  004610  012700  177776                MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
 6991  004614  005001                        CLR     R1              ;R1 CONTROLS DATA SHIFTING
 6992  004616  012777  177777  174674  1$:     MOV     #-1,@DRVWCR     ;DO ONE XFER
 6993  004624  012777  015364  174670        MOV     #DBUF,@DRVBAR   ;WRITE DATA WORD TO "DBUF"
 6994  004632  010077  174670                MOV     R0,@DRVDBR      ;SET UP DATA IN DBR
 6995  004636  012777  000103  174660        MOV     #103,@DRVCSR    ;SET IE, GO & FNCT1 (C1 CONTROL)
 6996  004644  052777  000400  174652        BIS     #400,@DRVCSR    ;SET CYCLE
 6997  004652  106427  000000                MTPS    #0              ;ENABLE THE INTR
 6998  004656  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 6999  004664  012737  001702  001124        MOV     #1702,$GDDAT    ;LD EXPECTED
 7000  004672  017737  174626  001126        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
 7001  004700  104005                        ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
 7002  004702  000442                        BR      4$              ;GO RESTORE VECTOR
 7003  004704  022626                2$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
 7004  004706  004537  010142                JSR     R5,CKSTAT       ;GO CHECK STATUS
 7005  004712  001702                        1702                    ;CSR STATUS EXPECTED
 7006  004714  000001                        1                       ;# OF XFERS
 7007  004716  104007                        ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
 7008                                                                ;CYCLE, READY, IE & FNCT 1
 7009  004720  000433                        BR      4$              ;GO RESTORE VECTOR
 7010  004722  104010                        ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
 7011  004724  000431                        BR      4$              ;GO RESTORE VECTOR
 7012  004726  104011                        ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
 7013  004730  000427                        BR      4$              ;GO RESTORE VECTOR
 7014  004732  012737  015364  001120        MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
 7015  004740  010037  001124                MOV     R0,$GDDAT       ;LD EXPECTED
 7016  004744  013737  015364  001126        MOV     DBUF,$BDDAT     ;GET DATA XFERED
 7017  004752  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
 7018  004760  001405                        BEQ     3$              ;BR IF SO
 7019  004762  013737  001526  001122        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
 7020  004770  104013                        ERROR   13              ;DATA ER - MEM CONTAINS WRONG DATA
```

```
7021  004772  000406                    BR      4$              ;GO RESTORE VECTOR
7022  004774  005100              3$:   COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
7023  004776  005101                    COM     R1              ;KEEP TRACK OF COMPLEMENT
7024  005000  001306                    BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
7025  005002  006300                    ASL     R0              ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
7026  005004  005200                    INC     R0              ;KEEP LSB SET
7027  005006  103703                    BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
7028  005010  004737  010116      4$:   JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7029
7030                                ;;**********************************************************************
7031                                ;*TEST 21        TEST 200 "DATI" NPR TRANSFERS (BURST MODE)
7032                                ;;**********************************************************************
7033  005014  000004              TST21:  SCOPE
7034  005016  012737  000010  001160      MOV     #10,$TIMES      ;;DO 10 ITERATIONS
7035  005024  004537  010076              JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
7036  005030  005126                      1$                      ;RETURN TO 1$ ON INTR
7037  005032  004737  010322              JSR     PC,LDBUF        ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
7038  005036  012777  177470  174454      MOV     #-200.,@DRVWCR  ;LOAD WC REG - WILL DO 200 XFERS
7039  005044  012777  015364  174450      MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
7040  005052  106427  000000              MTPS    #0              ;ENABLE THE INTR
7041  005056  012777  000101  174440      MOV     #101,@DRVCSR    ;SET IE & GO
7042  005064  052777  000400  174432      BIS     #400,@DRVCSR    ;SET CYCLE
7043  005072  013737  001524  001122      MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7044  005100  012737  000700  001124      MOV     #700,$GDDAT     ;LD EXPECTED
7045  005106  017737  174412  001126      MOV     @DRVCSR,$BDDAT  ;READ THE CSR
7046  005114  042777  000100  174402      BIC     #100,@DRVCSR    ;CLR INTR ENABLE
7047  005122  104005                      ERROR   5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
7048  005124  000434                      BR      2$              ;GO RESTORE VECTOR
7049  005126  022626              1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
7050  005130  004537  010142              JSR     R5,CKSTAT       ;GO CHECK STATUS
7051  005134  000700                      700                     ;CSR STATUS EXPECTED
7052  005136  000310                      200.                    ;# OF XFERS
7053  005140  104007                      ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
7054  005142  000425                      BR      2$              ;GO RESTORE VECTOR
7055  005144  104010                      ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
7056  005146  000423                      BR      2$              ;GO RESTORE VECTOR
7057  005150  104011                      ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
7058  005152  000421                      BR      2$              ;GO RESTORE VECTOR
7059  005154  012737  016202  001120      MOV     #DBUF+616,$GDADR  ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
7060  005162  012737  070707  001124      MOV     #70707,$GDDAT   ;LD EXPECTED
7061  005170  017737  174332  001126      MOV     @DRVDBR,$BDDAT  ;DBR SHOULD HAVE LAST DATUM
7062  005176  023737  001124  001126      CMP     $GDDAT,$BDDAT   ;CORRECT?
7063  005204  001404                      BEQ     2$              ;BR IF SO
7064  005206  013737  001526  001122      MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7065  005214  104012                      ERROR   12              ;DATA ER - DBR DID NOT CONTAIN EXPECTED LAST XFER
7066  005216  004737  010116      2$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7067
7068                                ;;**********************************************************************
7069                                ;*TEST 22        TEST 200 "DATO" NPR TRANSFERS (BURST MODE)
7070                                ;;**********************************************************************
7071  005222  000004              TST22:  SCOPE
7072  005224  012737  000010  001160      MOV     #10,$TIMES      ;;DO 10 ITERATIONS
7073  005232  004537  010076              JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
7074  005236  005336                      1$                      ;RETURN TO 1$ ON INTR
7075  005240  012777  177470  174252      MOV     #-200.,@DRVWCR  ;WORD WC REG - WILL DO 200 XFER'S
7076  005246  012777  015364  174246      MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
```

```
7077  005254  012777  177377  174244          MOV     #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
7078  005262  106427  000000                  MTPS    #0              ;ENABLE THE INTR
7079  005266  012777  000103  174230          MOV     #103,@DRVCSR    ;SET IE, FNCT 1 & GO
7080  005274  052777  000400  174222          BIS     #400,@DRVCSR    ;SET CYCLE
7081  005302  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7082  005310  012737  001702  001124          MOV     #1702,$GDDAT    ;LD EXPECTED
7083  005316  017737  174202  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
7084  005324  042777  000100  174172          BIC     #100,@DRVCSR    ;CLR INTR ENABLE
7085  005332  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
7086  005334  000416                          BR      2$              ;GO RESTORE VECTOR
7087  005336  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
7088  005340  004537  010142                  JSR     R5,CKSTAT       ;GO CHECK STATUS
7089  005344  001702                          1702                    ;CSR STATUS EXPECTED
7090  005346  000310                          200.                    ;# OF XFERS
7091  005350  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
7092                                                                  ;CYCLE, READY, IE & FNCT 1
7093  005352  000407                          BR      2$              ;GO RESTORE VECTOR
7094  005354  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
7095  005356  000405                          BR      2$              ;GO RESTORE VECTOR
7096  005360  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
7097  005362  000403                          BR      2$              ;GO RESTORE VECTOR
7098  005364  004737  010444                  JSR     PC,CKDAT        ;RETURN HERE IF OK - NOW GO CHECK DATA
7099  005370  104013                          ERROR   13              ;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
7100  005372  004737  010116          2$:     JSR     PC,RSTVEC       ;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
7101
7102                                          ;;**********************************************************************
7103                                          ;*TEST 23        TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
7104                                          ;;**********************************************************************
7105  005376  000004                  TST23:  SCOPE
7106  005400  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
7107  005406  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7108  005414  004537  010076                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
7109  005420  005530                          3$                      ;RETURN TO 3$ ON INTR
7110  005422  012700  000010                  MOV     #10,R0          ;DO EIGHT 200 WORD XFER'S
7111  005426  005037  001126                  CLR     $BDDAT          ;USR $BDDAT AS A COUNTER
7112  005432  012777  177470  174060  1$:     MOV     #-200.,@DRVWCR  ;DO 200 XFERS (DATI'S)
7113  005440  012777  015364  174054          MOV     #DBUF,@DRVBAR   ;FROM DBUF
7114  005446  106427  000000                  MTPS    #0              ;ALLOW AN INTR
7115  005452  012777  000101  174044          MOV     #101,@DRVCSR    ;SET IE & GO
7116  005460  052777  000400  174036          BIS     #400,@DRVCSR    ;SET CYCLE
7117  005466  005201                  5$:     INC     R1              ;INCREMENT R1 -DD001
7118  005470  001376                          BNE     5$              ;DD001
7119  005472  000240                          NOP
7120  005474  005237  001126          2$:     INC     $BDDAT          ;START COUNTING - SHOULD NEVER GET HERE
7121  005500  001375                          BNE     2$              ;UNTIL 64K
7122  005502  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
7123  005510  C17737  174010  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
7124  005516  042777  000100  174000          BIC     #100,@DRVCSR    ;CLR IE
7125  005524  104005                          ERROR   5               ;NO INTERRUPT ON 200 DATI'S
7126  005526  000407                          BR      4$              ;GO RESTORE VECTOR
7127  005530  022626                  3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
7128  005532  005300                          DEC     R0              ;DONE 8 TIMES?
7129  005534  001336                          BNE     1$              ;BR IF NOT
7130  005536  005737  001126                  TST     $BDDAT          ;SHOULD STILL BE ZERO
7131  005542  001401                          BEQ     4$              ;BR IF SO
7132  005544  104014                          ERROR   14              ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
```

```
7133  005546  004737  010116          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7134
7135                                  ;;*******************************************************************
7136                                  ;*TEST 24        TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
7137                                  ;;*******************************************************************
7138  005552  000004                  TST24:  SCOPE
7139  005554  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
7140  005562  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7141  005570  004537  010076                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
7142  005574  005702                          3$                      ;RETURN TO 3$ ON INTR
7143  005576  012700  000010                  MOV     #10,R0          ;DO EIGHT 200 WORD XFER'S
7144  005602  012737  000000  001126          MOV     #0,$BDDAT       ;USE $BDDAT AS A COUNTER
7145  005610  012777  177470  173702  1$:     MOV     #-200.,@DRVWCR  ;DO 200 XFERS (DATI'S)
7146  005616  012777  015364  173676          MOV     #DBUF,@DRVBAR   ;FROM DBUF
7147  005624  106427  000000                  MTPS    #0              ;ALLOW AN INTR
7148  005630  012777  000111  173666          MOV     #111,@DRVCSR    ;SET IE, FNCT3 & GO
7149  005636  052777  000400  173660          BIS     #400,@DRVCSR    ;SET CYCLE
7150  005644  000240                          NOP                     ;FREEBEE
7151  005646  005237  001126          2$:     INC     $BDDAT          ;START COUNTING
7152  005652  001375                          BNE     2$              ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
7153  005654  012737  004710  001124          MOV     #4710,$GDDAT    ;LD EXPECTED
7154  005662  017737  173636  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
7155  005670  042777  000100  173626          BIC     #100,@DRVCSR    ;CLR IE
7156  005676  104005                          ERROR   5               ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
7157  005700  000423                          BR      5$              ;GO RESTORE VECTOR
7158  005702  022626                  3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
7159  005704  005300                          DEC     R0              ;DONE 8 TIMES?
7160  005706  001340                          BNE     1$              ;BR IF NOT
7161  005710  022737  000000  001126          CMP     #0,$BDDAT       ;$BDDAT SHOULD HAVE BEEN COUNTED
7162  005716  103401                          BCS     4$              ;BR IF SO
7163  005720  104015                          ERROR   15              ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
7164  005722  017737  173576  001126  4$:     MOV     @DRVCSR,$BDDAT  ;READ STATUS
7165  005730  012737  004710  001124          MOV     #4710,$GDDAT    ;LD EXPECTED
7166  005736  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7167  005744  001401                          BEQ     5$              ;BR IF SO
7168  005746  104007                          ERROR   7               ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
7169  005750  004737  010116          5$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7170
7171                                  ;;*******************************************************************
7172                                  ;*TEST 25        TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
7173                                  ;;*******************************************************************
7174  005754  000004                  TST25:  SCOPE
7175  005756  012737  000200  001160          MOV     #200,$TIMES     ;;DO 200 ITERATIONS
7176  005764  004537  010076                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
7177  005770  006106                          2$                      ;RETURN TO 2$ ON INTR
7178  005772  004737  010374                  JSR     PC,LDBUF1       ;GO SET UP DBUF (SPECIAL COM PATTERN)
7179  005776  012737  011702  006114          MOV     #11702,3$       ;3$ CONTAINS EXPECTED STATUS
7180  006004  012737  000001  006116          MOV     #1,4$           ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
7181  006012  106427  000000          1$:     MTPS    #0              ;ALLOW INTR
7182  006016  012777  015364  173476          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
7183  006024  013777  006116  173466          MOV     4$,@DRVWCR      ;GET XFER #
7184  006032  005477  173462                  NEG     @DRVWCR         ;NEGATE FOR WC
7185  006036  012777  010101  173460          MOV     #10101,@DRVCSR  ;SET UP MAINT, IE & GO
7186  006044  052777  000400  173452          BIS     #400,@DRVCSR    ;SET CYCLE
7187  006052  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7188  006060  013737  006114  001124          MOV     3$,$GDDAT       ;LD EXPECTED
```

```
7189  006066  017737  173432  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
7190  006074  042777  000100  173422          BIC     #100,@DRVCSR    ;DISABLE IE
7191  006102  104005                          ERROR   5               ;NO INTR ON XFER (IN MAINT MD)
7192  006104  000440                          BR      6$              ;GO RESTORE VECTOR
7193  006106  022626                  2$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
7194  006110  004537  010142                  JSR     R5,CKSTAT       ;GO CHECK STATUS
7195  006114  011702                  3$:     11702                   ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
7196  006116  000001                  4$:     1                       ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
7197  006120  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - WILL EXPECT
7198                                                                  ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
7199                                                                  ;CYCLE, READY & IE
7200  006122  000431                          BR      6$              ;GO RESTORE VECTOR
7201  006124  104010                          ERROR   10              ;RETURN HERE IF WC ER - SHOULD BE 0
7202  006126  000427                          BR      6$              ;GO RESTORE VECTOR
7203  006130  104011                          ERROR   11              ;RETURN HERE IF BAR ER-
7204  006132  000425                          BR      6$              ;GO RESTORE VECTOR
7205  006134  062737  001002  006114          ADD     #1002,3$        ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
7206  006142  032737  020000  006114          BIT     #BIT13,3$       ;LOOK FOR OVERFLOW
7207  006150  001403                          BEQ     5$              ;BR IF NOT
7208  006152  012737  010700  006114          MOV     #10700,3$       ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
7209  006160  005237  006116          5$:     INC     4$              ;ADVANCE CURRENT XFER #
7210  006164  022737  000011  006116          CMP     #11,4$          ;HAVE 10 XFERS BEEN DONE
7211  006172  001307                          BNE     1$              ;BR IF NOT
7212  006174  004537  010512                  JSR     R5,CKDAT1       ;NOW GO CHECK DATA
7213  006200  000010                          10                      ;# OF XFER'S TO CHECK
7214  006202  104013                          ERROR   13              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
7215  006204  000240                          NOP                     ;RESTORE VECTOR NEXT
7216  006206  004737  010116          6$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7217
7218                                  ;;**************************************************************
7219                                  ;*TEST 26       TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
7220                                  ;;**************************************************************
7221  006212  000004                  TST26:  SCOPE
7222  006214  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
7223  006222  012700  000002                  MOV     #2,R0           ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
7224  006226  004537  010076          1$:     JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
7225  006232  006324                          2$                      ;RETURN TO 2$ ON TIMEOUT INTR
7226  006234  012777  177777  173256          MOV     #-1,@DRVWCR     ;SET UP FOR ONE XFER'S
7227  006242  012777  160000  173252          MOV     #160000,@DRVBAR ;SET UP CA TO 160000 (RESERVED)
7228  006250  106427  000000                  MTPS    #0              ;ALLOW INTR
7229  006254  012777  000161  173242          MOV     #161,@DRVCSR    ;SET IE, XAD 17,16 & GO
7230  006262  052777  000400  173234          BIS     #400,@DRVCSR    ;SET CYCLE
7231  006270  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS - SHOULD NEVER GET HERE
7232  006276  012737  140760  001124          MOV     #140760,$GDDAT  ;LD EXPECTED
7233  006304  017737  173214  001126          MOV     @DRVCSR,$BDDAT  ;GIVE THEM THE STATUS
7234  006312  042777  000100  173204          BIC     #100,@DRVCSR    ;CLR IE
7235  006320  104016                          ERROR   16              ;NEX FAILED TO CAUSE AN INTERRUPT
7236  006322  000521                          BR      7$              ;GO RESTORE VECTOR
7237  006324  022626                  2$:     CMP     (SP)+,(SP)+     ;SHOULD INTR RETURN HERE - FIX STACK
7238  006326  017737  173172  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
7239  006334  012737  140760  001124          MOV     #140760,$GDDAT  ;LD EXPECTED
7240  006342  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7241  006350  001405                          BEQ     3$              ;BR IF SO
7242  006352  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7243  006360  104016                          ERROR   16              ;STATUS ER - EXPECTED
7244                                                                  ;ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
```

```
7245  006362  000501                          BR      7$              ;GO RESTORE VECTOR
7246  006364  017737  173130  001126  3$:     MOV     @DRVWCR,$BDDAT  ;READ WORD COUNT
7247  006372  012737  177777  001124          MOV     #-1,$GDDAT      ;SHOULD STILL HAVE -1
7248  006400  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7249  006406  001405                          BEQ     4$              ;BR IF SO
7250  006410  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WCR ADRS
7251  006416  104010                          ERROR   10              ;WC INCREMENTED ON A TIMEOUT ER
7252  006420  000462                          BR      7$              ;GO RESTORE VECTOR
7253  006422  017737  173074  001126  4$:     MOV     @DRVBAR,$BDDAT  ;READ BUFFER ADRS
7254  006430  012737  160000  001124          MOV     #160000,$GDDAT  ;SHOULD NOT HAVE INCREMENTED
7255  006436  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7256  006444  001405                          BEQ     5$              ;BR IF SO
7257  006446  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
7258  006454  104011                          ERROR   11              ;BAR INCREMENTED ON A TIMEOUT ER
7259  006456  000443                          BR      7$              ;GO RESTORE VECTOR
7260  006460  005300                  5$:     DEC     R0              ;KEEP TRACK ON HOW TO CLR
7261  006462  001422                          BEQ     6$              ;BR IF CLR BY RESET
7262  006464  042777  040000  173032          BIC     #40000,@DRVCSR  ;WRITE NEX TO ZERO
7263  006472  017737  173026  001126          MOV     @DRVCSR,$BDDAT  ;READ CSR
7264  006500  012737  000760  001124          MOV     #760,$GDDAT     ;LD EXPECTED
7265  006506  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7266  006514  001644                          BEQ     1$              ;BR IF SO + REPEAT TEST FOR RESET TEST
7267  006516  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7268  006524  104016                          ERROR   16              ;'NEX' FAILED TO WRITE TO ZERO
7269  006526  000417                          BR      7$              ;GO RESTORE VECTOR
7270  006530  000005                  6$:     RESET                   ;ISSUE BUS RESET
7271  006532  017737  172766  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
7272  006540  012737  000200  001124          MOV     #200,$GDDAT     ;EXPECT ONLY READY
7273  006546  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
7274  006554  001404                          BEQ     7$              ;BR IF SO
7275  006556  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7276  006564  104016                          ERROR   16              ;RESET FAILED TO CLR 'NEX'
7277  006566  004737  010116          7$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7278                                  ;;**********************************************************************
7279                                  ;*TEST 27        TEST 200 NPR TRANSFERS IN MAINT MODE
7280                                  ;;**********************************************************************
7281  006572  000004                  TST27:  SCOPE
7282  006574  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
7283  006602  004537  010076                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
7284  006606  006720                          2$                      ;RETURN TO 2$ ON INTR
7285  006610  004737  010374                  JSR     PC,LDBUF1       ;GO SET UP DBUF (SPECIAL COM PATTERN)
7286  006614  012777  015364  172700          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
7287  006622  012777  177470  172670          MOV     #-200.,@DRVWCR  ;SET UP FOR 200 XFER'S
7288  006630  106427  000000                  MTPS    #0              ;ALLOW INTR
7289  006634  012777  010101  172662          MOV     #10101,@DRVCSR  ;SET MAINT, IE & GO
7290  006642  052777  000400  172654          BIS     #400,@DRVCSR    ;SET CYCLE
7291  006650  012737  000000  001126          MOV     #0,$BDDAT       ;SET UP A COUNTER
7292  006656  005237  001126          1$:     INC     $BDDAT          ;COUNT AWAY
7293  006662  001375                          BNE     1$              ;WAIT TILL DONE - SHOULD INTR BEFORE OVFLO
7294  006664  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7295  006672  012737  010700  001124          MOV     #10700,$GDDAT   ;LD EXPECTED
7296  006700  017737  172620  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
7297  006706  042777  000100  172610          BIC     #100,@DRVCSR    ;DISABLE IE
7298  006714  104005                          ERROR   5               ;NO INTR AFTER 200 MAINT MODE XFER'S
7299  006716  000420                          BR      3$              ;GO RESTORE VECTOR
7300  006720  022626                  2$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
```

```
7301  006722  004537  010142          JSR     R5,CKSTAT       ;GO CHECK STATUS
7302  006726  010700                  10700                   ;EXPECTED STATUS
7303  006730  000310                  200.                    ;# OF XFERS
7304  006732  104007                  ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
7305                                                          ;CYCLE, READY & IE
7306  006734  000411                  BR      3$              ;GO RESTORE VECTOR
7307  006736  104010                  ERROR   10              ;RETURN HERE IF WC ER - SHOULD BE 0
7308  006740  000407                  BR      3$              ;GO RESTORE VECTOR
7309  006742  104011                  ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
7310  006744  000405                  BR      3$              ;GO RESTORE VECTOR
7311  006746  004537  010512          JSR     R5,CKDAT1       ;RETURN HERE IF OK - NOW GO CHECK DATA
7312  006752  000310                  200.                    ;# OF XFER'S TO CHECK
7313  006754  104013                  ERROR   13              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
7314  006756  000240                  NOP                     ;RESTORE VECTOR NEXT
7315  006760  004737  010116  3$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7316                          ;;**********************************************************************
7317                          ;*TEST 30        TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
7318                          ;;**********************************************************************
7319  006764  000004          TST30:  SCOPE
7320  006766  012737  000005  001160  MOV     #5,$TIMES       ;;DO 5 ITERATIONS
7321  006774  004537  010076          JSR     R5,SETVEC       ;GO SET UP INTR RETURN
7322  007000  007134                  3$                      ;RETURN TO 3$ ON INTR
7323  007002  005037  001542          CLR     DBUFP           ;GET LOWEST BUFFER ADRS
7324  007006  062737  020000  001542  1$:     ADD     #20000,DBUFP    ;POINT TO NEXT 4K
7325  007014  023737  001540  001542  CMP     CORSZ,DBUFP     ;IS THE 4K THERE?
7326  007022  001465                  BEQ     4$              ;BR IF NOT
7327  007024  004737  010374          JSR     PC,LDBUF1       ;GO SET UP SPECIAL COMPEMENT PATTERN
7328  007030  013777  001542  172464  MOV     DBUFP,@DRVBAR   ;SET UP BUFFER ADRS
7329  007036  012777  177470  172454  MOV     #-200.,@DRVWCR  ;SET UP FOR 200 XFER'S
7330  007044  106427  000000          MTPS    #0              ;ALLOW INTR
7331  007050  012777  010101  172446  MOV     #10101,@DRVCSR  ;SET MAINT,IE & GO
7332  007056  052777  000400  172440  BIS     #400,@DRVCSR    ;SET CYCLE
7333  007064  012737  000000  001126  MOV     #0,$BDDAT       ;SET UP A COUNTER
7334  007072  005237  001126  2$:     INC     $BDDAT          ;COUNT AWAY
7335  007076  001375                  BNE     2$              ;SHOULD ALWAYS INTR FROM THIS LOOP
7336  007100  013737  001524  001122  MOV     DRVCSR,$BDADR   ;SET UP CSR ADRA
7337  007106  012737  010700  001124  MOV     #10700,$GDDAT   ;LD EXPECTED
7338  007114  017737  172404  001126  MOV     @DRVCSR,$BDDAT  ;READ CSR
7339  007122  042777  000100  172374  BIC     #100,@DRVCSR    ;DISABLE IE
7340  007130  104005                  ERROR   5               ;NO INTR AFTER 200 MAINT MODE XFER'S
7341  007132  000421                  BR      4$              ;GO RESTORE VECTOR
7342  007134  022626          3$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
7343  007136  004537  010142          JSR     R5,CKSTAT       ;GO CHECK STATUS
7344  007142  010700                  10700                   ;EXPECTED STATUS
7345  007144  000310                  200.                    ;# OF XFER'S
7346  007146  104007                  ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
7347                                                          ;CYCLE, READY & IE
7348  007150  000412                  BR      4$              ;GO RESTORE VECTOR
7349  007152  104010                  ERROR   10              ;RETURN HERE IF WC ER - SHOULD = 0
7350  007154  000410                  BR      4$              ;GO RESTORE VECTOR
7351  007156  104011                  ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUFP+620
7352  007160  000406                  BR      4$              ;GO RESTORE VECTOR
7353  007162  004537  010512          JSR     R5,CKDAT1       ;RETURN HERE IF OK - NOW GO CK DATA
7354  007166  000310                  200.                    ;# OF XFER'S TO CK
7355  007170  104013                  ERROR   13              ;RETURN HERE IF DATA ER
7356  007172  000401                  BR      4$              ;GO RESTORE VECTOR
```

```
 7357  007174  000704                          BR      1$                      ;TRY NEXT BANK
 7358  007176  012737  015364  001542  4$:     MOV     #DBUF,DBUFP             ;RESTORE BUFFER ADRS TO LOWEST 4K
 7359  007204  004737  010116                  JSR     PC,RSTVEC               ;GO RESTORE VECTOR
 7360                                          ;;**************************************************************************
 7361                                          ;*TEST 31        TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
 7362                                          ;;**************************************************************************
 7363  007210  000004                  TST31:  SCOPE
 7364  007212  105777  171722                  TSTB    @SWR                    ;KDF11-B?
 7365  007216  100507                          BMI     ALTERN                  ;IF YES,GO TO THE NEXT SUBTEST
 7366  007220  004537  010076          101$:   JSR     R5,SETVEC               ;GO SET UP INTR RETURN
 7367  007224  007330                          1$                              ;RETURN TO 1$ ON INTR
 7368  007226  012777  177777  172264          MOV     #-1,@DRVWCR             ;SET UP WC - 1 XFER
 7369  007234  013737  001150  001542          MOV     $TPS,DBUFP              ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
 7370  007242  013777  001150  172252          MOV     $TPS,@DRVBAR            ;SET UP BUFFER ADRS - FROM PRINTER CSR
 7371  007250  106427  000000                  MTPS    #0                      ;ALLOW INTR
 7372  007254  005077  172246                  CLR     @DRVDBR                 ;ZERO THE DBR
 7373  007260  012777  000161  172236          MOV     #161,@DRVCSR            ;SET IE, XAD17 & XAD16, & GO
 7374  007266  052777  000400  172230          BIS     #400,@DRVCSR            ;SET CYCLE
 7375  007274  013737  001524  001122          MOV     DRVCSR,$BDADR           ;SET UP CSR ADRS
 7376  007302  012737  000760  001124          MOV     #760,$GDDAT             ;LD EXPECTED STATUS
 7377  007310  017737  172210  001126          MOV     @DRVCSR,$BDDAT          ;READ THE CSR
 7378  007316  042777  000100  172200          BIC     #100,@DRVCSR            ;DISABLE IE
 7379  007324  104005                          ERROR   5                       ;NO INTR ON 1 WD XFER FROM XCSR
 7380  007326  000434                          BR      2$                      ;GO RESTORE VECTOR
 7381  007330  022626                  1$:     CMP     (SP)+,(SP)+             ;INTR RETURNS HERE - FIX STK SINCE NO RTI
 7382  007332  004537  010142                  JSR     R5,CKSTAT               ;GO CK STATUS
 7383  007336  000760                          760                             ;CSR EXPECTED STATUS
 7384  007340  000001                          1                               ;# OF XFER'S
 7385  007342  104007                          ERROR   7                       ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
 7386                                                                          ;XAD17 & XAD16, RDY & IE
 7387  007344  000425                          BR      2$                      ;GO RESTORE VECTOR
 7388  007346  104010                          ERROR   10                      ;RETURN HERE IF WC ER - EXPECTED 0
 7389  007350  000423                          BR      2$                      ;GO RESTORE VECTOR
 7390  007352  104011                          ERROR   11                      ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
 7391  007354  000421                          BR      2$                      ;GO RESTORE VECTOR
 7392  007356  017737  171566  001124          MOV     @$TPS,$GDDAT            ;RETURN HERE IF OK - GET XCSR CONTENTS
 7393  007364  017737  172136  001126          MOV     @DRVDBR,$BDDAT          ;READ DBR
 7394  007372  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;CORRECT?
 7395  007400  001407                          BEQ     2$                      ;BR IF SO
 7396  007402  013737  001526  001122          MOV     DRVDBR,$BDADR           ;SET UP DBR ADRS
 7397  007410  013737  001150  001120          MOV     $TPS,$GDADR             ;GET ADRS OF DATA (XCSR)
 7398  007416  104020                          ERROR   20                      ;DATA ER FROM TTY PRINTER CSR
 7399  007420  012737  015364  001542  2$:     MOV     #DBUF,DBUFP             ;RESTORE BUFFER ADRS TO LOWEST 4K
 7400  007426  004737  010116                  JSR     PC,RSTVEC               ;GO RESTORE VECTOR
 7401  007432  000137  007656                  JMP     NXDEV                   ;IGNORE NEXT SUBTEST
 7402
 7403                                          ;;**************************************************************************
 7404                                          ;IN CASE OF KDF11-B TEST SOME SPECIFIED I/O PAGE
 7405                                          ;;**************************************************************************
 7406  007436  005737  001544          ALTERN: TST     IOPAGE                  ;I/O TRANSFER DESIRED?
 7407  007442  001505                          BEQ     NXDEV                   ;IF NOT, SKIP THE TEST
 7408  007444  004537  010076                  JSR     R5,SETVEC               ;GO SET UP INTR RETURN
 7409  007450  007554                          1$                              ;RETURN TO 1$ ON INTR
 7410  007452  012777  177777  172040          MOV     #-1,@DRVWCR             ;SET UP WC - 1 XFER
 7411  007460  013737  001544  001542          MOV     IOPAGE,DBUFP            ;SET UP BUFFER ADRS TO I/O CSR ADRS
 7412  007466  013777  001544  172026          MOV     IOPAGE,@DRVBAR          ;SET UP BUFFER ADRS - FROM I/O CSR
```

```
7413  007474  106427  000000            MTPS    #0              ;ALLOW INTR
7414  007500  005077  172022            CLR     @DRVDBR         ;ZERO THE DBR
7415  007504  012777  000161  172012    MOV     #161,@DRVCSR    ;SET IE, XAD17 & XAD16, & GO
7416  007512  052777  000400  172004    BIS     #400,@DRVCSR    ;SET CYCLE
7417  007520  013737  001524  001122    MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
7418  007526  012737  000760  001124    MOV     #760,$GDDAT     ;LD EXPECTED STATUS
7419  007534  017737  171764  001126    MOV     @DRVCSR,$BDDAT  ;READ THE CSR
7420  007542  042777  000100  171754    BIC     #100,@DRVCSR    ;DISABLE IE
7421  007550  104005                    ERROR   5               ;NO INTR ON 1 WD XFER FROM XCSR
7422  007552  000434                    BR      2$              ;GO RESTORE VECTOR
7423  007554  022626            1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STK SINCE NO RTI
7424  007556  004537  010142            JSR     R5,CKSTAT       ;GO CK STATUS
7425  007562  000760                    760                     ;CSR EXPECTED STATUS
7426  007564  000001                    1                       ;# OF XFER'S
7427  007566  104007                    ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
7428                                                            ;XAD17 & XAD16, RDY & IE
7429  007570  000425                    BR      2$              ;GO RESTORE VECTOR
7430  007572  104010                    ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
7431  007574  000423                    BR      2$              ;GO RESTORE VECTOR
7432  007576  104011                    ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
7433  007600  000421                    BR      2$              ;GO RESTORE VECTOR
7434  007602  017737  171736  001124    MOV     @IOPAGE,$GDDAT  ;RETURN HERE IF OK - GET XCSR CONTENTS
7435  007610  017737  171712  001126    MOV     @DRVDBR,$BDDAT  ;READ DBR
7436  007616  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;CORRECT?
7437  007624  001407                    BEQ     2$              ;BR IF SO
7438  007626  013737  001526  001122    MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7439  007634  013737  001544  001120    MOV     IOPAGE,$GDADR   ;GET ADRS OF DATA (XCSR)
7440  007642  104020                    ERROR   20              ;DATA ER FROM TTY PRINTER CSR
7441  007644  012737  015364  001542  2$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
7442  007652  004737  010116            JSR     PC,RSTVEC       ;GO RESTORE VECTOR
7443
7444                                    ;;**************************************************************************
7445                                    ;DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
7446                                    ;;**************************************************************************
7447  007656  000004            NXDEV:  SCOPE
7448  007660  000241            NXDEV1: CLC                     ;CLR CARRY
7449  007662  006037  001536            ROR     DMAP            ;LOOK FOR NEXT
7450  007666  001432                    BEQ     $EOP            ;BR IF ALL TESTED
7451  007670  062737  000010  001520    ADD     #10,DRVWCR      ;OFFSET BASE BUS ADRS TO NEXT DRV11B
7452  007676  062737  000010  001522    ADD     #10,DRVBAR      ;
7453  007704  062737  000010  001524    ADD     #10,DRVCSR      ;
7454  007712  062737  000010  001526    ADD     #10,DRVDBR      ;
7455  007720  062737  000004  001530    ADD     #4,DRVCT0       ;OFFSET VECTOR ADRS TO NEXT
7456  007726  062737  000004  001532    ADD     #4,DRVCT2       ;
7457  007734  005237  001206            INC     $UNIT           ;COUNT DEVICE
7458  007740  032737  000001  001536    BIT     #1,DMAP         ;IS IT SELECTED?
7459  007746  001744                    BEQ     NXDEV1          ;BR IF NOT
7460  007750  000137  002372            JMP     RESTRT          ;TEST NEXT
7461                                    .SBTTL  END OF PASS ROUTINE
7462
7463                                    ;;**************************************************************************
7464                                    ;*INCREMENT THE PASS NUMBER ($PASS)
7465                                    ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
7466                                    ;*IF THERES A MONITOR GO TO IT
7467                                    ;*IF THERE ISN'T JUMP TO START1
7468
```

```
7469  007754                          $EOP:
7470  007754  000240                        NOP
7471  007756  005037  001102                CLR     $TSTNM          ;;ZERO THE TEST NUMBER
7472  007762  005037  001160                CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
7473  007766  005237  001202                INC     $PASS           ;;INCREMENT THE PASS NUMBER
7474  007772  042737  100000  001202        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
7475  010000  005327                        DEC     (PC)+           ;;LOOP?
7476  010002  000001                $EOPCT: .WORD   1
7477  010004  003022                        BGT     $DOAGN          ;;YES
7478  010006  012737                        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
7479  010010  000001                $ENDCT: .WORD   1
7480  010012  010002                        $EOPCT
7481  010014  104401  010061                TYPE    ,$ENDMG         ;;TYPE "END PASS #"
7482  010020  013746  001202                MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
7483  010024  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
7484  010026  104401  010056                TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
7485  010032  013700  000042        $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
7486  010036  001405                        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
7487  010040  000005                        RESET                   ;;CLEAR THE WORLD
7488  010042  004710                $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
7489  010044  000240                        NOP                     ;;SAVE ROOM
7490  010046  000240                        NOP                     ;;FOR
7491  010050  000240                        NOP                     ;;ACT11
7492  010052                        $DOAGN:
7493  010052  000137                        JMP     @(PC)+          ;;RETURN
7494  010054  002012                $RTNAD: .WORD   START1
7495  010056     377     377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
7496  010061     015  042412  042116 $ENDMG: .ASCIZ  <15><12>/END PASS #/
      010066  050040  051501  020123
      010074  000043
7497
```

```
                                .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
                                ;;*****************************************************************
010076  106427  000200          SETVEC: MTPS    #200            ;SET UP FOR NO INTERRUPT
010102  012577  171422                  MOV     (R5)+,@DRVCTO   ;SET UP INTR RETURN ADRS
010106  012777  000200  171416          MOV     #200,@DRVCT2    ;KEEP PRIORITY LEVEL AT TOP ON INTR
010114  000205                          RTS     R5              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE CLEARS THE DRV11B CSR - RESTORES THE DRV11B
                                ;INTERRUPT VECTOR TO A HALT - RAISES PRIORITY LEVEL
                                ;;*****************************************************************
010116  005077  171402          RSTVEC: CLR     @DRVCSR         ;CLR STATUS & CONTROL
010122  013777  001532  171400          MOV     DRVCT2,@DRVCTO  ;POINT VECTOR TO HALT
010130  005077  171376                  CLR     @DRVCT2         ;SET UP HALT
010134  106427  000200                  MTPS    #200            ;RAISE PRIORITY LEVEL
010140  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
                                ;;*****************************************************************
010142  017737  171356  001126  CKSTAT: MOV     @DRVCSR,$BDDAT  ;READ THE STATUS
010150  042777  000100  171346          BIC     #100,@DRVCSR    ;DISABLE THE IE BIT
010156  012537  001124                  MOV     (R5)+,$GDDAT    ;SET UP EXPECTED STATUS
010162  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
010170  001406                          BEQ     1$              ;BR IF SO
010172  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
010200  062705  000002                  ADD     #2,R5           ;POINT TO THE CSR ER
010204  000205                          RTS     R5              ;EXIT HERE ON STATUS ERROR
010206  017737  171306  001126  1$:     MOV     @DRVWCR,$BDDAT  ;GET WC
010214  001410                          BEQ     2$              ;BR IF ZERO
010216  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WCR ADRS
010224  005037  001124                  CLR     $GDDAT          ;EXPECTED 0
010230  062705  000006                  ADD     #6,R5           ;POINT TO THE WCR ER
010234  000205                          RTS     R5              ;EXIT HERE ON WCR ER
010236  011537  001124          2$:     MOV     (R5),$GDDAT     ;GET XFER #
010242  006337  001124                  ASL     $GDDAT          ;CONVERT TO WORD
010246  063737  001542  001124          ADD     DBUFP,$GDDAT    ;POINT TO LAST XFER +2
010254  017737  171242  001126          MOV     @DRVBAR,$BDDAT  ;GET BA
010262  042737  000001  001126          BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
010270  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
010276  001406                          BEQ     3$              ;BR IF SO
010300  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
010306  062705  000012                  ADD     #12,R5          ;POINT TO BAR ER
```

```
                        RTS     R5              ;EXIT HERE ON BAR ER
                3$:     ADD     #16,R5          ;ALL OK - POINT TO GOOD EXIT
                        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
                ;;*****************************************************************
LDBUF:  MOV     #DBUF,R3        ;GET BUFFER ADRS
1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
        COM     R4              ;MAKE INTO FLOATING 1
        CMP     #DBUF+616,R3    ;AT END OF BUFFER?
        BNE     4$              ;BR IF NOT
3$:     MOV     #70707,(R3)     ;LOAD LAST DATVAR (SPECIAL)
        RTS     PC              ;GET OUT
4$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 1)
        CMP     #DBUF+616,R3    ;AT END OF BUFFER?
        BEQ     3$              ;BR IF SO
        COM     R4              ;BACK TO FLOATING ZERO
        ASL     R4              ;SHIFT LEFT
        INC     R4              ;KEEP LSB SET
        BCC     1$              ;GO RESET FLOATING PATRN
        BR      2$              ;GO LOAD NEXT PATRN

                ;;*****************************************************************
                ;THIS ROUTINE LOADS 'DBUF' WITH A UNIQUE FLOATING ZERO/ONE PATTERN
                ;(177776,0,1,0,177775,0,2,0,177773,0,4,0,177767,0,10,0 ETC.)
                ;IT IS USED WITH MAINT BIT SET (DATI/DATO SEQUENCE) - 200 LOCS
                ;ARE LOADED WITH THIS PATTERN
                ;;*****************************************************************
LDBUF1: MOV     DBUFP,R3        ;GET BUFFER ADRS
        MOV     R3,R5           ;SAVE IN R5
        ADD     #620,R5         ;POINT TO END OF BUFFER
1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
        CLR     (R3)+           ;ZERO NEXT
        COM     R4              ;SET UP FLOATING 1
        MOV     R4,(R3)+        ;LOAD IT
        CLR     (R3)+           ;ZERO NEXT
        CMP     R5,R3           ;200 LOCS DONE?
        BNE     3$              ;BR IF NOT
        RTS     PC              ;GET OUT
3$:     COM     R4              ;BACK TO FLOATING ZERO
        ASL     R4              ;SHIFT LEFT
        INC     R4              ;KEEP LSB SET
        BCC     1$              ;GO RESET FLOATING PATRN
        BR      2$              ;GO FLOAT NEXT PATRN

                ;;*****************************************************************
                ;THIS ROUTINE CHECKS 200 LOCATIONS IN "DBUF" FOR GOOD TRANSFERED
                ;DATA (#177377) ON 'DATO' TRANSFERS - IF AN ERROR IS DETECTED
                ;THE RETURN IS TO CALL +2 - IF NO ERROR THE RETURN IS TO CALL +4
                ;;*****************************************************************
```

```
7611  010444  012701  015364          CKDAT:  MOV     #DBUF,R1        ;GET BUFFER ADRS
7612  010450  022721  177377          1$:     CMP     #177377,(R1)+   ;DATA OK?
7613  010454  001410                          BEQ     2$              ;BR IF SO
7614  010456  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7615  010464  014137  001126                  MOV     -(R1),$BDDAT    ;GET ACTUAL DATA XFERED
7616  010470  010137  001120                  MOV     R1,$GDADR       ;GET MEMORY ADRS
7617  010474  000207                          RTS     PC              ;RETURN TO ERROR
7618  010476  022701  016204          2$:     CMP     #DBUF+620,R1    ;AT END OF 'DBUF'?
7619  010502  001362                          BNE     1$              ;BR IF MORE
7620  010504  062716  000002                  ADD     #2,(SP)         ;ADJUST STACK FOR GOOD RETURN
7621  010510  000207                          RTS     PC              ;GET OUT
7622
7623                                  ;;**********************************************************************
7624                                  ;THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
7625                                  ;DATA (177776,177776,1,1,177775,177775,2,2,177773,177773,ETC.)
7626                                  ;ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED
7627                                  ;BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 -
7628                                  ;IF NO ERROR THE RETURN IS TO CALL +10
7629                                  ;;**********************************************************************
7630  010512  012500                  CKDAT1: MOV     (R5)+,R0        ;GET # OF CHECKS
7631  010514  013702  001542                  MOV     DBUFP,R2        ;GET BUFFER ADRS
7632  010520  012701  177776          1$:     MOV     #177776,R1      ;SET UP FLOATING ZERO PATRN
7633  010524  005003                          CLR     R3              ;R3 SAYS WHEN TO SHIFT PATRN
7634  010526  020122                  2$:     CMP     R1,(R2)+        ;DATA OK?
7635  010530  001010                          BNE     3$              ;BR IF NOT
7636  010532  020122                          CMP     R1,(R2)+        ;DATA WRITTEN OK?
7637  010534  001006                          BNE     3$              ;BR IF NOT
7638  010536  162700  000002                  SUB     #2,R0           ;ACCOUNT FOR TWO ADRS'S
7639  010542  003015                          BGT     4$              ;BR IF MORE
7640  010544  062705  000004                  ADD     #4,R5           ;ADJUST FOR GOOD RETURN
7641  010550  000205                          RTS     R5              ;EXIT
7642  010552  014237  001126          3$:     MOV     -(R2),$BDDAT    ;GET BAD DATA
7643  010556  010237  001120                  MOV     R2,$GDADR       ;GET MEM ADRS
7644  010562  010137  001124                  MOV     R1,$GDDAT       ;LD EXPECTED DATA
7645  010566  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
7646  010574  000205                          RTS     R5              ;RETURN TO ERROR
7647  010576  005101                  4$:     COM     R1              ;NOW EXPECT COMPLEMENT
7648  010600  005103                          COM     R3              ;TIME TO SHIFT?
7649  010602  001351                          BNE     2$              ;BR IF NOT
7650  010604  006301                          ASL     R1              ;SHIFT LEFT
7651  010606  005201                          INC     R1              ;KEEP LSB SET
7652  010610  103343                          BCC     1$              ;GO RESET FLOATING PATRN
7653  010612  000745                          BR      2$              ;DO NEXT
7654
```

```
                                        .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
                                ;*

010614  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010620  100002                          BPL     1$              ;;BR IF YES
010622  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
010624  000430                          BR      3$              ;;LEAVE
010626  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
010630  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010634  122737  000001  001214          CMPB    @APTENV,$ENV    ;;RUNNING IN APT MODE
010642  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
010644  132737  000100  001215          BITB    @APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
010652  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
010654  010037  010664                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
010660  004737  011156                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
010664  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
010666  132737  000040  001215  62$:    BITB    @APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
010674  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
010676  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010700  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010702  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010704  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
010706  062716  000002          3$:     ADD     @2,(SP)         ;;ADJUST RETURN PC
010712  000002                          RTI                     ;;RETURN
010714  122716  000011          4$:     CMPB    @HT,(SP)        ;;BRANCH IF <HT>
010720  001430                          BEQ     8$
010722  122716  000200                  CMPB    @CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
010726  001006                          BNE     5$
010730  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
010732  104401  001171                  TYPE    ,$CRLF          ;;TYPE A CR AND LF
010736  105037  011144                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
010742  000755                          BR      2$              ;;GET NEXT CHARACTER
010744  004737  011026          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
010750  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
010754  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
010756  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
```

```
7712                                                    ;;AND THE NULL CHAR.
7713  010762  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
7714  010766  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
7715  010770  004737  011026                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
7716  010774  105337  011144                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
7717  011000  000770                          BR      7$              ;;LOOP
7718
7719                                  ;HORIZONTAL TAB PROCESSOR
7720
7721  011002  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
7722  011006  004737  011026          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
7723  011012  132737  000007  011144          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
7724  011020  001372                          BNE     9$              ;;TAB STOP
7725  011022  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
7726  011024  000724                          BR      2$              ;;GET NEXT CHARACTER
7727  011026                          $TYPEC:
7728  011026  105777  170112                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
7729  011032  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
7730  011034  017746  170106                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
7731  011040  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
7732  011044  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
7733  011050  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
7734  011052                          101$:                                                           ;MJD001
7735  011052  105777  170066                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
7736  011056  100375                          BPL     101$                                            ;MJD001
7737  011060  117716  170062                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
7738  011064  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
7739  011070  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
7740  011074  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
7741  011076                          102$:                                                           ;MJD001
7742  011076  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
7743  011100                          10$:                                                            ;MJD001
7744  011100  105777  170044                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
7745  011104  100375                          BPL     10$                                             ;MJD001
7746  011106  116677  000002  170036          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.  ;MJD001
7747  011114  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
7748  011122  001003                          BNE     1$              ;;BRANCH IF NO
7749  011124  105037  011144                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
7750  011130  000406                          BR      $TYPEX          ;;EXIT
7751  011132  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
7752  011140  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
7753  011142  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
7754  011144  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
7755  011146  000207          $TYPEX: RTS     PC
7756
7757                                  .SBTTL  APT COMMUNICATIONS ROUTINE
7758
7759                                  ;;**************************************************************************
7760  011150  112737  000001  011414  $ATY1:   MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
7761  011156  112737  000001  011412  $ATY3:   MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
7762  011164  000403                          BR      $ATYC
7763  011166  112737  000001  011414  $ATY4:   MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
7764  011174                          $ATYC:
7765  011174  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7766  011176  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7767  011200  105737  011412                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
```

```
7768  011204  001450                        BEQ     5$                      ;;IF NOT:  BR
7769  011206  122737  000001  001214        CMPB    @APTENV,$ENV            ;;OPERATING UNDER APT?
7770  011214  001031                        BNE     3$                      ;;IF NOT:  BR
7771  011216  132737  000100  001215        BITB    @APTSPOOL,$ENVM         ;;SHOULD SPOOL MESSAGES?
7772  011224  001425                        BEQ     3$                      ;;IF NOT:  BR
7773  011226  017600  000004                MOV     @4(SP),R0               ;;GET MESSAGE ADDR.
7774  011232  062766  000002  000004        ADD     #2,4(SP)                        ;;BUMP RETURN ADDR.
7775  011240  005737  001174          1$:   TST     $MSGTYPE                ;;SEE IF DONE W/ LAST XMISSION?
7776  011244  001375                        BNE     1$                      ;;IF NOT:  WAIT
7777  011246  010037  001210                MOV     R0,$MSGAD               ;;PUT ADDR IN MAILBOX
7778  011252  105720                  2$:   TSTB    (R0)+                   ;;FIND END OF MESSAGE
7779  011254  001376                        BNE     2$
7780  011256  163700  001210                SUB     $MSGAD,R0               ;;SUB START OF MESSAGE
7781  011262  006200                        ASR     R0                      ;;GET MESSAGE LNGTH IN WORDS
7782  011264  010037  001212                MOV     R0,$MSGLGT              ;;PUT LENGTH IN MAILBOX
7783  011270  012737  000004  001174        MOV     #4,$MSGTYPE             ;;TELL APT TO TAKE MSG.
7784  011276  000413                        BR      5$
7785  011300  017637  000004  011324  3$:   MOV     @4(SP),4$               ;;PUT MSG ADDR IN JSR LINKAGE
7786  011306  062766  000002  000004        ADD     #2,4(SP)                        ;;BUMP RETURN ADDRESS
7787  011314  013746  177776                MOV     177776,-(SP)            ;;PUSH 177776 ON STACK
7788  011320  004737  010614                JSR     PC,$TYPE                ;;CALL TYPE MACRO
7789  011324  000000                  4$:   .WORD   0
7790  011326                           5$:
7791  011326  105737  011414         10$:   TSTB    $FFLG                   ;;SHOULD REPORT FATAL ERROR?
7792  011332  001416                        BEQ     12$                     ;;IF NOT:  BR
7793  011334  005737  001214                TST     $ENV                    ;;RUNNING UNDER APT?
7794  011340  001413                        BEQ     12$                     ;;IF NOT:  BR
7795  011342  005737  001174         11$:   TST     $MSGTYPE                ;;FINISHED LAST MESSAGE?
7796  011346  001375                        BNE     11$                     ;;IF NOT:  WAIT
7797  011350  017637  000004  001176        MOV     @4(SP),$FATAL           ;;GET ERROR #
7798  011356  062766  000002  000004        ADD     #2,4(SP)                        ;;BUMP RETURN ADDR.
7799  011364  005237  001174                INC     $MSGTYPE                ;;TELL APT TO TAKE ERROR
7800  011370  105037  011414         12$:   CLRB    $FFLG                   ;;CLEAR FATAL FLAG
7801  011374  105037  011413                CLRB    $LFLG                   ;;CLEAR LOG FLAG
7802  011400  105037  011412                CLRB    $MFLG                   ;;CLEAR MESSAGE FLAG
7803  011404  012601                        MOV     (SP)+,R1                ;;POP STACK INTO R1
7804  011406  012600                        MOV     (SP)+,R0                ;;POP STACK INTO R0
7805  011410  000207                        RTS     PC                      ;;RETURN
7806  011412     000           $MFLG:  .BYTE   0                       ;;MESSG. FLAG
7807  011413     000           $LFLG:  .BYTE   0                       ;;LOG FLAG
7808  011414     000           $FFLG:  .BYTE   0                       ;;FATAL FLAG
7809          011416                  .EVEN
7810          000200          APTSIZE=200
7811          000001          APTENV=001
7812          000100          APTSPOOL=100
7813          000040          APTCSUP=040
7814                          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
7815
7816                          ;;**********************************************************************
7817                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
7818                          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
7819                          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
7820                          ;*CALL:
7821                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
7822                          ;*      TYPOS                   ;;CALL FOR TYPEOUT
7823                          ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
```

```
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                              ;;1=TYPE LEADING ZEROS
                                        ;*                              ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
011416  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
011422  116637  000001  011641          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
011430  112637  011643                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
011434  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
011440  000406                          BR      $TYPON
011442  112737  000001  011641  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
011450  112737  000006  011643          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
011456  112737  000005  011640  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
011464  010346                          MOV     R3,-(SP)        ;;SAVE R3
011466  010446                          MOV     R4,-(SP)        ;;SAVE R4
011470  010546                          MOV     R5,-(SP)        ;;SAVE R5
011472  113704  011643                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
011476  005404                          NEG     R4
011500  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
011504  110437  011642                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
011510  113704  011641                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
011514  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
011520  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
011522  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
011524  000404                          BR      3$              ;;GO DO MSB
011526  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
011530  006105                          ROL     R5
011532  006105                          ROL     R5
011534  010503                          MOV     R5,R3
011536  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
011540  105337  011642                  DECB    $OMODE          ;;TYPE THIS DIGIT?
011544  100016                          BPL     7$              ;;BR IF NO
011546  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
011552  001002                          BNE     4$              ;;TEST FOR 0
011554  005704                          TST     R4              ;;SUPPRESS THIS 0?
011556  001403                          BEQ     5$              ;;BR IF YES
011560  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
011562  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
011566  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
011572  110337  011636                  MOVB    R3,8$           ;;SAVE FOR TYPING
011576  104401  011636                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
011602  105337  011640          7$:     DECB    $OCNT           ;;COUNT BY 1
011606  003347                          BGT     2$              ;;BR IF MORE TO DO
011610  002402                          BLT     6$              ;;BR IF DONE
011612  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
011614  000744                          BR      2$              ;;GO DO THE LAST DIGIT
```

```
 7880  011616  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
 7881  011620  012604                          MOV     (SP)+,R4        ;;RESTORE R4
 7882  011622  012603                          MOV     (SP)+,R3        ;;RESTORE R3
 7883  011624  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 7884  011632  012616                          MOV     (SP)+,(SP)
 7885  011634  000002                          RTI                     ;;RETURN
 7886  011636     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 7887  011637     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 7888  011640     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
 7889  011641     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
 7890  011642  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 7891                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 7892
 7893                                  ;;**************************************************************************
 7894                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 7895                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 7896                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 7897                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 7898                                  ;*REPLACED WITH SPACES.
 7899                                  ;*CALL:
 7900                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 7901                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
 7902
 7903  011644                          $TYPDS:
 7904  011644  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 7905  011646  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 7906  011650  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 7907  011652  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 7908  011654  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 7909  011656  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 7910  011662  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 7911  011666  100004                          BPL     1$              ;;BR IF INPUT IS POS.
 7912  011670  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 7913  011672  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 7914  011700  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 7915  011702  012703  012060                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 7916  011706  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 7917  011712  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 7918  011714  016001  012050                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 7919  011720  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
 7920  011722  002402                          BLT     4$              ;;BR IF DONE
 7921  011724  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
 7922  011726  000774                          BR      3$
 7923  011730  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
 7924  011732  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
 7925  011734  001002                          BNE     5$              ;;FALL THROUGH IF 0
 7926  011736  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
 7927  011740  100407                          BMI     7$              ;;BR IF YES
 7928  011742  106316                  5$:     ASLB    (SP)            ;;MSD?
 7929  011744  103003                          BCC     6$              ;;BR IF NO
 7930  011746  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
 7931  011754  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
 7932  011760  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
 7933  011764  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
 7934  011766  005720                          TST     (R0)+           ;;JUST INCREMENTING
 7935  011770  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
```

```
 7936  011774  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
 7937  011776  003002                        BGT     8$              ;;GO TO EXIT
 7938  012000  010502                        MOV     R5,R2           ;;GET THE LSD
 7939  012002  000764                        BR      6$              ;;GO CHANGE TO ASCII
 7940  012004  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
 7941  012006  100003                        BPL     9$              ;;BR IF NO
 7942  012010  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
 7943  012016  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
 7944  012020  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
 7945  012022  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
 7946  012024  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
 7947  012026  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
 7948  012030  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 7949  012032  104401  012060                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
 7950  012036  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
 7951  012044  012616                        MOV     (SP)+,(SP)
 7952  012046  000002                        RTI                     ;;RETURN TO USER
 7953  012050  023420                $DTBL:  10000.
 7954  012052  001750                        1000.
 7955  012054  000144                        100.
 7956  012056  000012                        10.
 7957  012060  000004                $DBLK:  .BLKW   4
 7958                                 .SBTTL  ERROR HANDLER ROUTINE
 7959
 7960                                 ;;**************************************************************************
 7961                                 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 7962                                 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 7963                                 ;*AND GO TO SWRCK ON ERROR
 7964                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 7965                                 ;*SW15=1         HALT ON ERROR
 7966                                 ;*SW13=1         INHIBIT ERROR TYPEOUTS
 7967                                 ;*SW10=1         BELL ON ERROR
 7968                                 ;*SW09=1         LOOP ON ERROR
 7969                                 ;*CALL
 7970                                 ;*       ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 7971
 7972  012070                $ERROR:
 7973  012070  104407                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 7974  012072  104407                        CKSWR           ;GO LOOK FOR SWR CHANGE
 7975  012074  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 7976  012100  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 7977  012102  013777  001102  167032        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 7978  012110  032777  002000  167022        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 7979  012116  001402                        BEQ     1$              ;;NO - SKIP
 7980  012120  104401  001164                TYPE    ,$BELL          ;;RING BELL
 7981  012124  005237  001112        1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 7982  012130  011637  001116                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 7983  012134  162737  000002  001116        SUB     #2,$ERRPC
 7984  012142  117737  166750  001114        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 7985  012150  032777  020000  166762        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 7986  012156  001004                        BNE     20$             ;;SKIP TYPEOUTS
 7987  012160  004737  012260                JSR     PC,SWRCK        ;;GO TO USER ERROR ROUTINE
 7988  012164  104401  001171                TYPE    ,$CRLF
 7989  012170                        20$:
 7990  012170  122737  000001  001214        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 7991  012176  001007                        BNE     2$              ;;NO,SKIP APT ERROR REPORT
```

```
 7992  012200  113737  001114  012212          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 7993  012206  004737  011166                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 7994  012212    000                   21$:    .BYTE   0
 7995  012213    000                           .BYTE   0
 7996  012214  000777                  22$:    BR      22$             ;;APT ERROR LOOP
 7997  012216  005777  166716          2$:     TST     @SWR            ;;HALT ON ERROR
 7998  012222  100002                          BPL     3$              ;;SKIP IF CONTINUE
 7999  012224  000000                          HALT                    ;;HALT ON ERROR!
 8000  012226  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 8001  012230  032777  001000  166702  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 8002  012236  001402                          BEQ     4$              ;;BR IF NO
 8003  012240  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 8004  012244  005737  001162          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 8005  012250  001402                          BEQ     5$              ;;BR IF NONE
 8006  012252  013716  001162                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 8007  012256                          5$:
 8008  012256  000002                          RTI                     ;;RETURN
 8009                                  ;;**************************************************************************
 8010                                  ;GO TYPE ERROR
 8011                                  ;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
 8012                                  ;;**************************************************************************
 8013  012260  113737  001102  001534  SWRCK:  MOVB    $TSTNM,TSTNUM   ;SET UP TEST # ON ER
 8014  012266  004737  012276                  JSR     PC,$ERRTYP      ;GO TYPE ERROR
 8015  012272  104407                          CKSWR                   ;GO LOOK FOR SWR CHANGE
 8016  012274  000207                          RTS     PC              ;RETURN TO ERROR HANDLER
 8017                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
 8018
 8019                                  ;;**************************************************************************
 8020                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
 8021                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
 8022                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
 8023
 8024  012276                          $ERRTYP:
 8025  012276  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 8026  012302  010046                          MOV     R0,-(SP)        ;;SAVE R0
 8027  012304  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
 8028  012306  153700  001114                  BISB    @#$ITEMB,R0
 8029  012312  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
 8030                                                                  ;;TYPE THE PC OF THE ERROR
 8031  012314  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
 8032                                                                  ;;ERROR ADDRESS
 8033  012320  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 8034  012322  000426                          BR      6$              ;;GET OUT
 8035  012324  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
 8036  012326  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
 8037  012330  006300                          ASL     R0
 8038  012332  006300                          ASL     R0
 8039  012334  062700  001320                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
 8040  012340  012037  012350                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
 8041  012344  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
 8042  012346  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
 8043  012350  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
 8044  012352  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 8045  012356  012037  012366          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
 8046  012362  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
 8047  012364  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
```

```
 8048  012366  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
 8049  012370  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 8050  012374  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
 8051  012376  001004                          BNE     7$              ;;GO TYPE THE DATA
 8052  012400  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
 8053  012402  104401  001171                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 8054  012406  000207                          RTS     PC              ;;RETURN
 8055  012410                          7$:
 8056  012410  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
 8057  012412  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 8058  012414  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
 8059  012416  001770                          BEQ     6$              ;;BR IF NO
 8060  012420  104401  012426                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
 8061  012424  000771                          BR      7$              ;;LOOP
 8062  012426  020040     000          8$:     .ASCIZ  / /             ;;TWO(2) SPACES
 8063          012432                          .EVEN
 8064                                  .SBTTL  SCOPE HANDLER ROUTINE
 8065
 8066                                  ;;**************************************************************************
 8067                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 8068                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 8069                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 8070                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 8071                                  ;*SW14=1         LOOP ON TEST
 8072                                  ;*SW11=1         INHIBIT ITERATIONS
 8073                                  ;*SW09=1         LOOP ON ERROR
 8074                                  ;*SW08=1         LOOP ON TEST IN SWR<5:0>
 8075                                  ;*CALL
 8076                                  ;*      SCOPE           ;;SCOPE=IOT
 8077
 8078  012432                          $SCOPE:
 8079  012432  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 8080  012434  104407                          CKSWR                   ;GO LOOK FOR SWR CHANGE
 8081  012436  032777  040000  166474  1$:     BIT     @#BIT14,@SWR    ;;LOOP ON PRESENT TEST?
 8082  012444  001117                          BNE     $OVER           ;;YES IF SW14=1
 8083                                  ;#####START OF CODE FOR THE XOR TESTER#####
 8084  012446  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 8085                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 8086  012450  013746  000004                  MOV     @#@#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 8087  012454  012737  012474  000004          MOV     #5$,@#@#ERRVEC  ;;SET FOR TIMEOUT
 8088  012462  005737  177060                  TST     @#@#177060      ;;TIME OUT ON XOR?
 8089  012466  012637  000004                  MOV     (SP)+,@#@#ERRVEC ;;RESTORE THE ERROR VECTOR
 8090  012472  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
 8091  012474  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 8092  012476  012637  000004                  MOV     (SP)+,@#@#ERRVEC ;;RESTORE THE ERROR VECTOR
 8093  012502  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
 8094  012504                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 8095  012504  032777  000400  166426          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 8096  012512  001407                          BEQ     2$              ;;BR IF NO
 8097  012514  017746  166420                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
 8098  012520  042716  000300                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
 8099  012524  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
 8100  012530  001465                          BEQ     $OVER           ;;BR IF YES
 8101  012532  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 8102  012536  001421                          BEQ     3$              ;;BR IF NO
 8103  012540  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
```

```
8104  012546  101015                        BHI     3$              ;;BR IF NO
8105  012550  032777  001000  166362        BIT     @BIT09,@SWR     ;;LOOP ON ERROR?
8106  012556  001404                        BEQ     4$              ;;BR IF NO
8107  012560  013737  001110  001106  7$:   MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
8108  012566  000446                        BR      $OVER
8109  012570  105037  001103          4$:   CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
8110  012574  005037  001160                CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
8111  012600  000415                        BR      1$              ;;ESCAPE TO THE NEXT TEST
8112  012602  032777  004000  166330  3$:   BIT     @BIT11,@SWR     ;;INHIBIT ITERATIONS?
8113  012610  001011                        BNE     1$              ;;BR IF YES
8114  012612  005737  001202                TST     $PASS           ;;IF FIRST PASS OF PROGRAM
8115  012616  001406                        BEQ     1$              ;;      INHIBIT ITERATIONS
8116  012620  005237  001104                INC     $ICNT           ;;INCREMENT ITERATION COUNT
8117  012624  023737  001160  001104        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
8118  012632  002024                        BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
8119  012634  012737  0000#1  001104  1$:   MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
8120  012642  013737  012720  001160        MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
8121  012650  105237  001102          $SVLAD: INCB  $TSTNM          ;;COUNT TEST NUMBERS
8122  012654  113737  001102  001200        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
8123  012662  011637  001106                MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
8124  012666  011637  001110                MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
8125  012672  005037  001162                CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
8126  012676  112737  000001  001115        MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
8127  012704  013777  001102  166230  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
8128  012712  013716  001106                MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
8129  012716  000002                        RTI                     ;;FIXES PS
8130  012720  003720                $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
8131                                 .SBTTL  TTY INPUT ROUTINE
8132
8133                                 ;;**************************************************************************
8134                                 .ENABL  LSB
8135
8136                                 ;;**************************************************************************
8137                                 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
8138                                 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
8139                                 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
8140                                 ;*WHEN OPERATING IN TTY FLAG MODE.
8141  012722  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
8142  012730  001074                        BNE     15$             ;;BRANCH IF NO
8143  012732  105777  166206                TSTB    @$TKS           ;;CHAR THERE?
8144  012736  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
8145  012740  117746  166202                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
8146  012744  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
8147  012750  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
8148  012754  001062                        BNE     15$             ;;NO, RETURN TO USER
8149  012756  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
8150  012764  001456                        BEQ     15$             ;;BRANCH IF YES
8151
8152  012766  104401  013457                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
8153  012772  104401  013464          $GTSWR: TYPE  ,$MSWR          ;;TYPE CURRENT CONTENTS
8154  012776  013746  000176                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
8155  013002  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
8156  013004  104401  013475                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
8157  013010  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
8158  013012  005046                        CLR     -(SP)           ;;THE NEW SWR
8159  013014  105777  166124          7$:   TSTB    @$TKS           ;;CHAR THERE?
```

```
8160  013020  100375                    BPL     7$              ;;IF NOT TRY AGAIN
8161
8162  013022  117746  166120            MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
8163  013026  042716  177600            BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
8164
8165
8166
8167  013032  021627  000025    9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
8168  013036  001005                    BNE     10$             ;;BRANCH IF NOT
8169  013040  104401  013452            TYPE    .$CNTLU         ;;YES, ECHO CONTROL-U (^U)
8170  013044  062706  000006    20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
8171  013050  000757                    BR      19$             ;;LET'S TRY IT AGAIN
8172
8173
8174  013052  021627  000015    10$:    CMP     (SP),#15        ;;IS IT A <CR>?
8175  013056  001022                    BNE     16$             ;;BRANCH IF NO
8176  013060  005766  000004            TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
8177  013064  001403                    BEQ     11$             ;;BRANCH IF YES
8178  013066  016677  000002  166044    MOV     2(SP),@SWR      ;;SAVE NEW SWR
8179  013074  062706  000006    11$:    ADD     #6,SP           ;;CLEAR UP STACK
8180  013100  104401  001171    14$:    TYPE    .$CRLF          ;;ECHO <CR> AND <LF>
8181  013104  123727  001135  000001    CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
8182  013112  001003                    BNE     15$             ;;BRANCH IF NOT
8183  013114  012777  000100  166022    MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
8184  013122  000002    15$:    RTI                             ;;RETURN
8185  013124  004737  011026    16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
8186  013130  021627  000060            CMP     (SP),#60        ;;CHAR < 0?
8187  013134  002420                    BLT     18$             ;;BRANCH IF YES
8188  013136  021627  000067            CMP     (SP),#67        ;;CHAR > 7?
8189  013142  003015                    BGT     18$             ;;BRANCH IF YES
8190  013144  042726  000060            BIC     #60,(SP)+       ;;STRIP-OFF ASCII
8191  013150  005766  000002            TST     2(SP)           ;;IS THIS THE FIRST CHAR
8192  013154  001403                    BEQ     17$             ;;BRANCH IF YES
8193  013156  006316                    ASL     (SP)            ;;NO, SHIFT PRESENT
8194  013160  006316                    ASL     (SP)            ;;   CHAR OVER TO MAKE
8195  013162  006316                    ASL     (SP)            ;;   ROOM FOR NEW ONE.
8196  013164  005266  000002    17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
8197  013170  056616  177776            BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
8198  013174  000707                    BR      7$              ;;GET THE NEXT ONE
8199  013176  104401  001170    18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
8200  013202  000720                    BR      20$             ;;SIMULATE CONTROL-U
8201                          .DSABL  LSB
8202
8203
8204                          ;;**********************************************************************
8205                          ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
8206                          ;*CALL:
8207                          ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
8208                          ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
8209                          ;*                              ;;WITH PARITY BIT STRIPPED OFF
8210                          ;
8211
8212  013204  011646    $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
8213  013206  016666  000004  000002    MOV     4(SP),2(SP)     ;;SAVE THE PS
8214  013214  105777  165724    1$:     TSTB    @$TKS           ;;WAIT FOR
8215  013220  100375                    BPL     1$              ;;A CHARACTER
```

```
 8216  013222  117766  165720  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
 8217  013230  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
 8218  013236  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
 8219  013244  001013                          BNE     3$              ;;BRANCH IF NO
 8220  013246  105777  165672          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
 8221  013252  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
 8222  013254  117746  165666                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
 8223  013260  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 8224  013264  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
 8225  013270  001366                          BNE     2$              ;;IF NOT DISCARD IT
 8226  013272  000750                          BR      1$              ;;YES, RESUME
 8227  013274  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
 8228  013302  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
 8229  013304  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
 8230  013312  002407                          BLT     4$              ;;BRANCH IF YES
 8231  013314  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
 8232  013322  003003                          BGT     4$              ;;BRANCH IF YES
 8233  013324  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
 8234  013332  000002                  4$:     RTI                     ;;GO BACK TO USER
 8235                                          ;;**************************************************************
 8236                                          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
 8237                                          ;*CALL:
 8238                                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
 8239                                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
 8240                                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
 8241
 8242  013334  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
 8243  013336  012703  013442          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
 8244  013342  022703  013452          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
 8245  013346  101405                          BLOS    4$              ;;BR IF YES
 8246  013350  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
 8247  013352  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
 8248  013354  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
 8249  013360  001003                          BNE     3$              ;;SKIP IF NOT
 8250  013362  104401  001170          4$:     TYPE    ,$QUES          ;;TYPE A '?'
 8251  013366  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
 8252  013370  111337  013440          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
 8253  013374  104401  013440                  TYPE    ,9$
 8254  013400  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
 8255  013404  001356                          BNE     2$              ;;LOOP IF NOT RETURN
 8256  013406  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
 8257  013412  104401  001172                  TYPE    ,$LF            ;;TYPE A LINE FEED
 8258  013416  012603                          MOV     (SP)+,R3        ;;RESTORE R3
 8259  013420  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
 8260  013422  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
 8261  013430  012766  013442  000004          MOV     #$TTYIN,4(SP)
 8262  013436  000002                          RTI                     ;;RETURN
 8263  013440     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
 8264  013441     000                          .BYTE   0               ;;TERMINATOR
 8265  013442  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
 8266  013452  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
 8267  013457     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
 8268  013464  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
       013472  020075     000
 8269
 8270  013475     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
```

```
      013502  036440  000040
 8271
 8272                                 .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 8273
 8274                                 ;;**********************************************************************
 8275                                 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 8276                                 ;*CHANGE IT TO BINARY.
 8277                                 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
 8278                                 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
 8279                                 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
 8280                                 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
 8281                                 ;*CALL:
 8282                                 ;*      RDOCT                   ;;READ AN OCTAL NUMBER
 8283                                 ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
 8284                                 ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 8285
 8286  013506  011646                 $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 8287  013510  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
 8288  013516  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 8289  013520  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8290  013522  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 8291  013524  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
 8292  013526  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 8293  013530  010037  013634                  MOV     R0,5$           ;;AND SAVE IT
 8294  013534  005001                          CLR     R1              ;;CLEAR DATA WORD
 8295  013536  005002                          CLR     R2
 8296  013540  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 8297  013542  001420                          BEQ     3$              ;;IF ZERO GET OUT
 8298  013544  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
 8299  013550  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
 8300  013552  122716  000067                  CMPB    #'7,(SP)
 8301  013556  002423                          BLT     4$
 8302  013560  006301                          ASL     R1              ;;*2
 8303  013562  006102                          ROL     R2
 8304  013564  006301                          ASL     R1              ;;*4
 8305  013566  006102                          ROL     R2
 8306  013570  006301                          ASL     R1              ;;*8
 8307  013572  006102                          ROL     R2
 8308  013574  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 8309  013600  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 8310  013602  000756                          BR      2$              ;;LOOP
 8311  013604  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 8312  013606  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
 8313  013612  010237  013644                  MOV     R2,$HIOCT
 8314  013616  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 8315  013620  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 8316  013622  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 8317  013624  000002                          RTI                     ;;RETURN
 8318  013626  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
 8319  013630  105010                          CLRB    (R0)            ;;SET A TERMINATOR
 8320  013632  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
 8321  013634  000000                  5$:     .WORD   0
 8322  013636  104401  001170                  TYPE    .$QUES          ;;"?" "CR" & "LF"
 8323  013642  000730                          BR      1$              ;;TRY AGAIN
 8324  013644  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
 8325                                 .SBTTL  POWER DOWN AND UP ROUTINES
```

```
                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
013646  012737  014012  000024  $PWRDN: MOV     @#ILLUP,@#PWRVEC ;;SET FOR FAST UP
013654  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
013662  010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013664  010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013666  010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013670  010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
013672  010446                       MOV     R4,-(SP)        ;;PUSH R4 ON STACK
013674  010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
013676  017746  165236               MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
013702  010637  014016               MOV     SP,$SAVR6       ;;SAVE SP
013706  012737  013720  000024       MOV     @#PWRUP,@#PWRVEC ;;SET UP VECTOR
013714  000000                       HALT
013716  000776                       BR      .-2             ;;HANG UP
                                ;;*****************************************************************
                                ;POWER UP ROUTINE
013720  012737  014012  000024  $PWRUP: MOV     @#ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
013726  013706  014016               MOV     $SAVR6,SP       ;;GET SP
013732  005037  014016               CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
013736  005237  014016         1$:   INC     $SAVR6          ;;WAIT FOR THE INC
013742  001375                       BNE     1$              ;;OF  WORD
013744  012677  165170               MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
013750  012605                       MOV     (SP)+,R5        ;;POP STACK INTO R5
013752  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
013754  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
013756  012602                       MOV     (SP)+,R2        ;;POP STACK INTO R2
013760  012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
013762  012600                       MOV     (SP)+,R0        ;;POP STACK INTO R0
013764  012737  013646  000024       MOV     @#PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
013772  012737  000340  000026       MOV     #340,@#PWRVEC+2 ;;PRIO:7
014000  104401                       TYPE                    ;;REPORT THE POWER FAILURE
014002  014020           $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
014004  012716                   MOV     (PC)+,(SP)      ;;RESTART AT RESTRT
014006  002372           $PWRAD: .WORD   RESTRT          ;;RESTART ADDRESS
014010  000002                   RTI
014012  000000           $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
014014  000776                   BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
014016  000000           $SAVR6: 0                       ;;PUT THE SP HERE
014020  005015  042522  052123 PWRMSG: .ASCIZ  <15><12>/RESTARTED FROM PWR FAIL/
014026  051101  042524  020104
014034  051106  046517  050040
014042  051127  043040  044501
014050  000114

                                .EVEN
                                .SBTTL  TRAP DECODER

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
```

```
                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                        ;*GO TO THAT ROUTINE.

014052 010046           $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
014054 016600 000002            MOV     2(SP),R0        ;;GET TRAP ADDRESS
014060 005740                   TST     -(R0)           ;;BACKUP BY 2
014062 111000                   MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
014064 006300                   ASL     R0              ;;POSITION FOR INDEXING
014066 016000 014106            MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
014072 000200                   RTS     R0              ;;GO TO ROUTINE

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

014074 011646           $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
014076 016666 000004 000002     MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
014104 000002                   RTI                     ;;RESTORE THE PSW

                        .SBTTL  TRAP TABLE

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

                        ;       ROUTINE
                        ;       -------
014106 014074           $TRPAD: .WORD   $TRAP2
014110 010614                   $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
014112 011442                   $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
014114 011416                   $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
014116 011456                   $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
014120 011644                   $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

014122 012772                   $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

014124 012722                   $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
014126 013204                   $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
014130 013334                   $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
014132 013506                   $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

                                .SBTTL  ASCII MESSAGES
014134 042522 020107 044524 EM1:    .ASCIZ  /REG TIMEOUT ER/
014142 042515 052517 020124
014150 051105    000

014153    122 043505 051040 EM2:    .ASCIZ  'REG READ/WRITE ER'
014160 040505 027504 051127
014166 052111 020105 051105
014174    000

014175    102 051525 051040 EM3:    .ASCIZ  /BUS RESET ER/
014202 051505 052105 042440
014210 000122
```

```
8427
8428  014212 047106 052103 041040 EM4:    .ASCIZ  /FNCT BITS FAILED TO SET STAT BITS/
      014220 052111 020123 040506
      014226 046111 042105 052040
      014234 020117 042523 020124
      014242 052123 052101 041040
      014250 052111 000123
8429
8430
8431
8432
8433
8434  014254 042522 042101 020131 EM5:    .ASCIZ  /READY INTR FAILURE/
      014262 047111 051124 043040
      014270 044501 052514 042522
      014276    000
8435
8436
8437
8438  014277    122 040505 054504 EM6:    .ASCIZ  /READY CLR OR SET ER/
      014304 041440 051114 047440
      014312 020122 042523 020124
      014320 051105    000
8439
8440
8441
8442  014323    123 040524 052524 EM7:    .ASCIZ  /STATUS ER ON XFER/
      014330 020123 051105 047440
      014336 020116 043130 051105
      014344    000
8443
8444
8445
8446  014345    127 051117 020104 EM10:   .ASCIZ  /WORD COUNT ER ON XFER/
      014352 047503 047125 020124
      014360 051105 047440 020116
      014366 043130 051105    000
8447
8448
8449
8450  014373    102 043125 042506 EM11:   .ASCIZ  /BUFFER ADRS ER ON XFER/
      014400 020122 042101 051522
      014406 042440 020122 047117
      014414 054040 042506 000122
8451
8452
8453
8454  014422 040504 040524 042440 EM12:   .ASCIZ  /DATA ER FROM MEM/
      014430 020122 051106 046517
      014436 046440 046505    000
8455
8456
8457  014443    104 052101 020101 EM13:   .ASCIZ  /DATA ER TO MEM/
      014450 051105 052040 020117
      014456 042515 000115
8458
```

```
 8459
 8460  014462  044523  043516  042514  EM14:   .ASCIZ  /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
       014470  041440  041531  042514
       014476  047440  043106  042040
       014504  042111  047040  052117
       014512  046040  041517  020113
       014520  052517  020124  050103
       014526  000125
 8461
 8462
 8463
 8464
 8465
 8466
 8467  014530  044523  043516  042514  EM15:   .ASCIZ  /SINGLE CYCLE ON LOCKED OUT CPU/
       014536  041440  041531  042514
       014544  047440  020116  047514
       014552  045503  042105  047440
       014560  052125  041440  052520
       014566     000
 8468
 8469
 8470
 8471
 8472
 8473  014567     015  050012  042514  WARN:   .ASCII  <15><12>/PLEASE DISABLE "REV11" MEMORY REFRESH OPTION/
       014574  051501  020105  044504
       014602  040523  046102  020105
       014610  051042  053105  030461
       014616  020042  042515  047515
       014624  054522  051040  043105
       014632  042522  044123  047440
       014640  052120  047511     116
 8474
 8475
 8476
 8477
 8478
 8479
 8480
 8481  014645     015  040412  042116          .ASCIZ  <15><12>/AND ENABLE PROCESSOR MEMORY REFRESH   /
       014652  042440  040516  046102
       014660  020105  051120  041517
       014666  051505  047523  020122
       014674  042515  047515  054522
       014702  051040  043105  042522
       014710  044123  020040  000040
 8482
 8483
 8484
 8485
 8486
 8487
 8488  014716  042516  020130  047514  EM16:   .ASCIZ  /NEX LOGIC ER/
       014724  044507  020103  051105
       014732     000
```

```
 8489
 8490
 8491  014733   103  041531  042514  EM17:   .ASCIZ  /CYCLE FAILED TO CLK DBR (IN)/
       014740  043040  044501  042514
       014746  020104  047524  041440
       014754  045514  042040  051102
       014762  024040  047111  000051
 8492
 8493
 8494
 8495
 8496  014770  040504  040524  042440  EM20:   .ASCIZ  "DATA ER FROM I/O PAGE (XCSR)"
       014776  020122  051106  046517
       015004  044440  047457  050040
       015012  043501  020105  054050
       015020  051503  024522     000
 8497
 8498
 8499
 8500
 8501  015025   105  051122  041520  DH1:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  EXPCT   RCVD/
       015032  020040  052040  052123
       015040  052516  020115  041040
       015046  051525  042101  020122
       015054  042440  050130  052103
       015062  020040  051040  053103
       015070  000104
 8502
 8503
 8504
 8505
 8506
 8507
 8508  015072  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  ADRS    EXPCT   RCVD/
       015100  020040  051524  047124
       015106  046525  020040  052502
       015114  040523  051104  020040
       015122  042101  051522  020040
       015130  020040  054105  041520
       015136  020124  020040  041522
       015144  042126     000
 8509
 8510
 8511
 8512
 8513
 8514
 8515
 8516  015147   105  051122  041520  DH3:    .ASCIZ  /ERRPC   TSTNUM  BUSADR/
       015154  020040  052040  052123
       015162  052516  020115  041040
       015170  051525  042101  000122
 8517
 8518
 8519
 8520
```

```
8521
8522  015176  001116  001534  001122  DT1:    .EVEN
      015204  001124  001126  000000          $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
8523
8524  015212  001116  001534  001122  DT2:    $ERRPC,TSTNUM,$BDADR,$GDADR,$GDDAT,$BDDAT,0
      015220  001120  001124  001126
      015226  000000
8525
8526
8527  015230  001116  001534  001122  DT3:    $ERRPC,TSTNUM,$BDADR,0
      015236  000000
8528
8529  015240  005015  051511  052040  KDF11B: .ASCIZ  <15><12>/IS THE PROCESSOR KDF11-B?  /
      015246  042510  050040  047522
      015254  042503  051523  051117
      015262  045440  043104  030461
      015270  041055  020077  000040
8530
8531
8532
8533
8534  015276  005015  051124  047101  IOTEST: .ASCIZ  <15><12>.TRANSFERS FOR I/O PAGE?  .
      015304  043123  051105  020123
      015312  047506  020122  027511
      015320  020117  040520  042507
      015326  020077  000040
8535
8536
8537
8538
8539  015332  005015  042101  051104  IOADR:  .ASCIZ  <15><12>\ADDRESS OF I/O PAGE?  \
      015340  051505  020123  043117
      015346  044440  047457  050040
      015354  043501  037505  020040
      015362     000
8540
8541
8542
8543
8544          015364                          .EVEN
8545                                  ;;**************************************************************************
8546                                  ;'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
8547                                  ;NPR OPERATIONS - IT IS 200 WORDS LONG
8548                                  ;;**************************************************************************
8549  015364  000000                  DBUF:   0               ;1ST ADRS OF DATA BUFFER
8550          000001                          .END
```

```
ABASE = 172410          6183#   6375
ACDW1 = 000000          6218#   6377
ACDW2 = 000000          6219#   6378
ACPUOP= 000000          6204#   6349
ADDW0 = 000000          6220#   6379
ADDW1 = 000000          6221#   6380
ADDW10= 000000          6230#   6389
ADDW11= 000000          6231#   6390
ADDW12= 000000          6232#   6391
ADDW13= 000000          6233#   6392
ADDW14= 000000          6234#   6393
ADDW15= 000000          6235#   6394
ADDW2 = 000000          6222#   6381
ADDW3 = 000000          6223#   6382
ADDW4 = 000000          6224#   6383
ADDW5 = 000000          6225#   6384
ADDW6 = 000000          6226#   6385
ADDW7 = 000000          6227#   6386
ADDW8 = 000000          6228#   6387
ADDW9 = 000000          6229#   6388
ADEVCT= 000000          6196#   6340
ADEVM = 000001          6186#   6376
AENV  = 000000          6200#   6345
AENVM = 000000          6201#   6346
AFATAL= 000000          6193#   6337
ALTERN  007436          7365    7406#
AMADR1= 000000          6207#   6362
AMADR2= 000000          6210#   6366
AMADR3= 000000          6213#   6369
AMADR4= 000000          6216#   6372
AMAMS1= 000000          6205#   6356
AMAMS2= 000000          6208#   6364
AMAMS3= 000000          6211#   6367
AMAMS4= 000000          6214#   6370
AMSGAD= 000000          6198#   6342
AMSGLG= 000000          6199#   6343
AMSGTY= 000000          6192#   6336
AMTYP1= 000000          6206#   6357
AMTYP2= 000000          6209#   6365
AMTYP3= 000000          6212#   6368
AMTYP4= 000000          6215#   6371
APASS = 000000          6195#   6339
APTCSU= 000040          7692    7813#
APTENV= 000001          7685    7769    7811#   7990
APTSIZ= 000200          6569    7810#
APTSPO= 000100          7687    7771    7812#
ASWREG= 000000          6202#   6347
ATESTN= 000000          6194#   6338
AUNIT = 000000          6197#   6341
AUSWR = 000000          6203#   6348
AVECT1= 000124          6185#   6373
AVECT2= 000000          6217#   6374
BIT0  = 000001          6168#
BIT00 = 000001          6158#   6168    6698    6703    6762    7550
BIT01 = 000002          6157#   6167
BIT02 = 000004          6156#   6166
```

```
BIT03 = 000010          6155#  6165
BIT04 = 000020          6154#  6164
BIT05 = 000040          6153#  6163
BIT06 = 000100          6152#  6162
BIT07 = 000200          6151#  6161  6614
BIT08 = 000400          6150#  6160  8095
BIT09 = 001000          6149#  6159  8001  8105
BIT1  = 000002          6167#
BIT10 = 002000          6148#  7978
BIT11 = 004000          6147#  8112
BIT12 = 010000          6146#
BIT13 = 020000          6145#  7206  7985
BIT14 = 040000          6144#  8081
BIT15 = 100000          6143#
BIT2  = 000004          6166#
BIT3  = 000010          6165#
BIT4  = 000020          6164#
BIT5  = 000040          6163#
BIT6  = 000100          6162#  6862
BIT7  = 000200          6161#
BIT8  = 000400          6160#
BIT9  = 001000          6159#
BPTVEC= 000014          6175#
CKDAT   010444          7098   7611#
CKDAT1  010512          7212   7311   7353   7630#
CKSTAT  010142          6958   7004   7050   7088   7194   7301   7343   7382   7424   7532#
CKSWR = 104407          6051#  7973   7974   8000   8015   8079   8080
CORSZ   001540          6525#  6617   6628*  7325
CORSZR  002270          6618   6623#
CR    = 000015          6083#  7747
CRLF  = 000200          6084#  6603   7702
DBUF    015364          6526   6630   6946   6947*  6967   6993   7014   7016   7039   7059   7076   7113   7146
                        7182   7286   7358   7399   7441   7565   7569   7574   7611   7618   8549#
DBUFP   001542          6526#  6630*  7323*  7324*  7325   7328   7358*  7369*  7399*  7411*  7441*  7548   7588
                        7631
DDISP = 177570          6090#  6315   6557
DH1     015025          6416   6422   6428   6434   6440   6446   6451   6457   6463   6493   6499   8501#
DH2     015072          6469   6475   6504   8508#
DH3     015147          6481   6487   8516#
DISPLA  001142          6315#  6557*  6565*  7977*  8127*
DISPRE  000174          6247#  6565
DMAP    001536          6524#  6632*  6633*  6635   7449*  7458
DRVBAR  001522          6512#  6693   6696*  6697   6754*  6761   6764   6902   6904*  6905   6910   6946*  6993*
                        7039*  7076*  7113*  7146*  7182*  7227*  7253   7257   7286*  7328*  7370*  7412*  7452*
                        7549   7553
DRVCSR  001524          6513#  6780   6784*  6785   6795   6805   6807*  6809   6818   6836   6839*  6840   6857
                        6862*  6864   6868   6872   6883   6885*  6886   6890*  6892   6901*  6909*  6914*  6923*
                        6929*  6948*  6949*  6951   6953   6954*  6995*  6996*  6998   7000   7041*  7042*  7043
                        7045   7046*  7079*  7080*  7081   7083   7084*  7107   7115*  7116*  7123   7124*  7140
                        7148*  7149*  7154   7155*  7164   7185*  7186*  7187   7189   7190*  7229*  7230*  7231
                        7233   7234*  7238   7242   7262*  7263   7267   7271   7275   7289*  7290*  7294   7296
                        7297*  7331*  7332*  7336   7338   7339*  7373*  7374*  7375   7377   7378*  7415*  7416*
                        7417   7419   7420*  7453*  7520*  7532   7533*  7537
DRVCTO  001530          6518#  6579   6856*  6861*  7455*  7512*  7521*
DRVCT2  001532          6519#  6584   7456*  7513*  7521   7522*
DRVDBR  001526          6514#  6577   6716   6719*  6720   6735   6755*  6766   6768   6920   6922*  6924*  6925
```

```
                    6931    6969    6972    6994*   7019    7061    7064    7077*   7372*   7393    7396    7414*   7435
                    7438    7454*   7614    7645
DRVWCR  001520      6511#   6573    6643    6656    6673    6676*   6677    6753*   6757    6759    6945*   6992*   7038*
                    7075*   7112*   7145*   7183*   7184*   7226*   7246    7250    7287*   7329*   7368*   7410*   7451*
                    7540    7542
DSWR  = 177570      6089#   6314    6556
DT1     015176      6417    6423    6429    6435    6441    6447    6452    6458    6464    6494    6500    8522#
DT2     015212      6470    6476    6505    8524#
DT3     015230      6482    6488    8527#
EMTVEC= 000030      6178#   6541*   6542*
EM1     014134      6415    8418#
EM10    014345      6456    8446#
EM11    014373      6462    8450#
EM12    014422      6468    8454#
EM13    014443      6474    8457#
EM14    014462      6480    8460#
EM15    014530      6486    8467#
EM16    014716      6492    8488#
EM17    014733      6498    8491#
EM2     014153      6421    8421#
EM20    014770      6503    8496#
EM3     014175      6427    8425#
EM4     014212      6433    8428#
EM5     014254      6439    8434#
EM6     014277      6445    8438#
EM7     014323      6450    8442#
ERRVEC= 000004      6171#   6554    6555*   6566*   6624*   6629*   6655*   6666*   8086    8087*   8089*   8092*
GTSWR = 104406      6050#   6598
HT    = 000011      6081#   7700
IOADR   015332      8539#
IOPAGE  001544      6527#   7406    7411    7412    7434    7439
IOTEST  015276      8534#
IOTVEC= 000020      6176#   6539*   6540*
KDF     001546      6528#
KDF11B  015240      8529#
LDBUF   010322      7037    7565#
LDBUF1  010374      7178    7285    7327    7588#
LF    = 000012      6082#   7751
MFPT  = 000007      6044#   6613
MTPS  = 106427      6187#
NXDEV   007656      7401    7407    7447#
NXDEV1  007660      6637    7448#   7459
PIRQ  = 177772      6088#
PIRQVE= 000240      6182#
PR0   = 000000      6105#
PR1   = 000040      6106#
PR2   = 000100      6107#
PR3   = 000140      6108#
PR4   = 000200      6109#
PR5   = 000240      6110#
PR6   = 000300      6111#
PR7   = 000340      6112#
PS    = 177776      6085#   6086
PSW   = 177776      6086#
PWRMSG  014020      8360    8367#
PWRVEC= 000024      6177#   6545*   6546*   8329*   8330*   8339*   8345*   8357*   8358*
```

```
RDCHR = 104410          6052#   8246
RDLIN = 104411          6053#   8291
RDOCT = 104412          6054#
RESTRT  002372          6634    6636    6638#   7460    8362
RESVEC= 000010          6172#
RSTVEC  010116          6798    6876    6981    7028    7066    7100    7133    7169    7216    7277    7315    7359    7400
                        7442    7520#
SETUP2  002022          6575#   6578
SETUP3  002052          6582#   6585
SETVEC  010076          6777    6941    6988    7035    7073    7108    7141    7176    7224    7283    7321    7366    7408
                        7511#
STACK = 001100          6076#   6537    6639
START   001550          6250    6530#
START1  002012          6573#   7494
STKLMT= 177774          6087#
SWR     001140          6314#   6535    6556*   6558    6564*   6571*   6596    6614*   7364    7978    7985    7997    8001
                        8081    8095    8097    8105    8112    8141    8178*   8337    8350*
SWRCK   012260          7987    8013#
SWREG   000176          6248#   6564    6596    8141    8154
SW0   = 000001          6140#
SW00  = 000001          6130#   6140
SW01  = 000002          6129#   6139
SW02  = 000004          6128#   6138
SW03  = 000010          6127#   6137
SW04  = 000020          6126#   6136
SW05  = 000040          6125#   6135
SW06  = 000100          6124#   6134
SW07  = 000200          6123#   6133
SW08  = 000400          6122#   6132
SW09  = 001000          6121#   6131
SW1   = 000002          6139#
SW10  = 002000          6120#
SW11  = 004000          6119#
SW12  = 010000          6118#
SW13  = 020000          6117#
SW14  = 040000          6116#
SW15  = 100000          6115#
SW2   = 000004          6138#
SW3   = 000010          6137#
SW4   = 000020          6136#
SW5   = 000040          6135#
SW6   = 000100          6134#
SW7   = 000200          6133#
SW8   = 000400          6132#
SW9   = 001000          6131#
TBITVE= 000014          6173#
TKVEC = 000060          6180#
TMAIN = 177522          6188#
TPVEC = 000064          6181#
TRAP  = 104400          6184#
TRAPVE= 000034          6179#   6543*   6544*
TRTVEC= 000014          6174#
TSTNUM  001534          6523#   8013*   8522    8524    8527
TST1    002426          6648#
TST10   003426          6803#
TST11   003502          6811    6817#
```

```
TST12   003564          6834#
TST13   003662          6852#
TST14   004022          6881#
TST15   004124          6894    6900#
TST16   004226          6919#
TST17   004342          6933    6939#
TST2    002536          6671#
TST20   004572          6986#
TST21   005014          7033#
TST22   005222          7071#
TST23   005376          7105#
TST24   005552          7138#
TST25   005754          7174#
TST26   006212          7221#
TST27   006572          7281#
TST3    002634          6691#
TST30   006764          7319#
TST31   007210          7363#
TST4    002752          6707    6714#
TST5    003050          6734#
TST6    003130          6744    6750#
TST7    003256          6767    6774#
TYPDS = 104405          6049#   7483
TYPE  = 104401          6045#   6590    6619    7481    7484    7705    7874    7949    7980    7988    8025    8042    8044
                        8047    8049    8053    8060    8152    8153    8156    8169    8180    8199    8250    8253    8257
                        8320    8322    8359
TYPOC = 104402          6046#   8033    8057    8155
TYPON = 104404          6048#
TYPOS = 104403          6047#
WARN    014567          6619    8473#
$APTHD  001000          6274    6280#
$ATYC   011174          7762    7764#
$ATY1   011150          7760#
$ATY3   011156          7690    7761#
$ATY4   011166          7763#   7993
$AUTOB  001134          6311#   6600*   8149
$BASE   001250          6375#   6574
$BDADR  001122          6306#   6658*   6673*   6693*   6716*   6736*   6759*   6764*   6768*   6780*   6805*   6819*   6836*
                        6857*   6883*   6902*   6920*   6951*   6972*   6998*   7019*   7043*   7064*   7081*   7107*   7140*
                        7187*   7231*   7242*   7250*   7257*   7267*   7275*   7294*   7336*   7375*   7396*   7417*   7438*
                        7537*   7542*   7553*   7614*   7645*   8522    8524    8527
$BDDAT  001126          6308#   6654*   6677*   6678    6697*   6698*   6699    6720*   6721    6739*   6740    6741    6742*
                        6743    6757*   6761*   6762*   6766*   6785*   6786*   6787    6795*   6796*   6809*   6810    6822*
                        6823    6824    6825*   6826    6840*   6841*   6842    6864*   6868*   6872*   6873    6886*   6887
                        6892*   6893    6905*   6906    6910*   6925*   6926    6931*   6932    6953*   6969*   6970    7000*
                        7016*   7017    7045*   7061*   7062    7083*   7111*   7120*   7123*   7130    7144*   7151*   7154*
                        7161    7164*   7166    7189*   7233*   7238*   7240    7246*   7248    7253*   7255    7263*   7265
                        7271*   7273    7291*   7292*   7296*   7333*   7334*   7338*   7377*   7393*   7394    7419*   7435*
                        7436    7532*   7535    7540*   7549*   7550*   7551    7615*   7642*   8522    8524
$BELL   001164          6326#   7980
$CDW1   001254          6377#
$CDW2   001256          6378#
$CHARC  011144          7706*   7716*   7723    7749*   7754#
$CKSWR  012722          8141#   8412
$CMTAG  001100          6294#   6533
$CNTLG  013457          8152    8267#
```

```
$CNTLU  013452          8169    8266#
$CPUOP  001222          6349#
$CRLF   001171          6328#   7705    7988    8025    8044    8049    8053    8180
$DBLK   012060          7915    7949    7957#
$DDW0   001260          6379#   6643*
$DDW1   001262          6380#
$DDW10  001304          6389#
$DDW11  001306          6390#
$DDW12  001310          6391#
$DDW13  001312          6392#
$DDW14  001314          6393#
$DDW15  001316          6394#
$DDW2   001264          6381#
$DDW3   001266          6382#
$DDW4   001270          6383#
$DDW5   001272          6384#
$DDW6   001274          6385#
$DDW7   001276          6386#
$DDW8   001300          6387#
$DDW9   001302          6388#
$DEVCT  001204          6340#   6640*
$DEVM   001252          6376#   6632
$DOAGN  010052          7477    7486    7492#
$DTBL   012050          7918    7953#
$ENDAD  010042          6260    7488#
$ENDCT  010010          7479#
$ENDMG  010061          7481    7496#
$ENULL  010056          7484    7495#
$ENV    001214          6345#   6594    7685    7769    7793    7990
$ENVM   001215          6346#   6569    7687    7692    7771
$EOP    007754          7450    7469#
$EOPCT  010002          7476#   7480
$ERFLG  001103          6297#   7975*   8101    8103    8109*
$ERMAX  001115          6303#   6549*   8103    8126*
$ERROR  012070          6541    7972#
$ERRPC  001116          6304#   7982*   7983*   7984    8031    8522    8524    8527
$ERRTB  001320          6413#   8039
$ERRTY  012276          8014    8024#
$ERTTL  001112          6301#   7981*
$ESCAP  001162          6325#   6548*   8004    8006    8125*
$ETABL  001214          6344#
$ETEND  001320          6286    6397#
$FATAL  001176          6337#   7797*
$FFLG   011414          7760*   7763*   7791    7800*   7808#
$FILLC  001156          6322#   7709
$FILLS  001155          6321#
$GDADR  001120          6305#   6967*   7014*   7059*   7397*   7439*   7616*   7643*   8524
$GDDAT  001124          6307#   6653*   6675*   6676    6678    6681*   6684*   6685*   6695*   6696    6699    6702*   6703*
                        6706*   6708*   6718*   6719    6721    6724*   6727*   6728*   6738*   6743    6752*   6782*   6783*
                        6787    6794*   6806*   6810    6821*   6826    6837*   6842    6845*   6858*   6873    6884*   6887
                        6891*   6893    6903*   6906    6912*   6921*   6922    6926    6930*   6932    6952*   6968*   6970
                        6999*   7015*   7017    7044*   7060*   7062    7082*   7122*   7153*   7165*   7166    7188*   7232*
                        7239*   7240    7247*   7248    7254*   7255    7264*   7265    7272*   7273    7295*   7337*   7376*
                        7392*   7394    7418*   7434*   7436    7534*   7535    7543*   7546*   7547*   7548*   7551    7644*
                        8522    8524
$GET42  010032          7485#
```

```
$GTSWR  012772          8153#   8410
$HIBTS  001000          6281#
$HIOCT  013644          8313*   8324#
$ICNT   001104          6298#   8116*   8117    8119*
$ILLUP  014012          8329    8345    8364#
$INTAG  001135          6312#   8181
$ITEMB  001114          6302#   7984*   7992    8028
$LF     001172          6329#   8257
$LFLG   011413          7801*   7807#
$LPADR  001106          6299#   6550*   6649*   8107*   8123*   8128
$LPERR  001110          6300#   6551*   6652*   6672*   6692*   6715*   6779*   6835*   6940*   6987*   8003    8107    8124*
$MADR1  001226          6362#
$MADR2  001232          6366#
$MADR3  001236          6369#
$MADR4  001242          6372#
$MAIL   001174          6282    6286    6335#
$MAMS1  001224          6356#
$MAMS2  001230          6364#
$MAMS3  001234          6367#
$MAMS4  001240          6370#
$MBADR  001002          6282#
$MFLG   011412          7761*   7767    7802*   7806#
$MNEW   013475          8156    8270#
$MSGAD  001210          6342#   7777*   7780
$MSGLG  0C1212          6343#   7782*
$MSGTY  001174          6336#   7775    7783*   7795    7799*
$MSWR   013464          8153    8268#
$MTYP1  001225          6357#
$MTYP2  001231          6365#
$MTYP3  001235          6368#
$MTYP4  001241          6371#
$MXCNT  012720          8120    8130#
$NULL   001154          6320#   7711
$OCNT   011640          7846*   7875*   7888#
$OMODE  011642          7841*   7845*   7850    7853*   7864*   7890#
$OVER   012704          8082    8100    8108    8118    8127#
$PASS   001202          6339#   6568*   7473*   7474*   7482    8114
$PASTM  001006          6284#
$PWRAD  014006          8362#
$PWRDN  013646          6545    8329#   8357
$PWRMG  014002          8360#
$PWRUP  013720          8339    8345#
$QUES   001170          6327#   8199    8250    8322
$RDCHR  013204          8212#   8413
$RDLIN  013334          8242#   8414
$RDOCT  013506          8286#   8415
$RTNAD  010054          7494#
$SAVR6  014016          8338*   8346    8347*   8348*   8366#
$SCOPE  012432          6539    8078#
$SETUP= 000137          6024#
$STUP = 177777          6024#
$SVLAD  012650          8090    8121#
$SVPC = 000106          6258#   6263
$SWR  = 167400          6058#   6059#
$SWREG  001216          6347#   6571
$SWRMK= 000300          6060#   8098
```

```
$TESTN  001200          6338#  6650*  8122*
$TIMES  001160          6324#  6547*  6751*  6775*  6804*  6853*  7034*  7072*  7106*  7139*  7175*  7222*  7282*
                        7320*  7472*  8110*  8117   8120*
$TKB    001146          6317#  7730   7737   8145   8162   8216   8222
$TKS    001144          6316#  7728   7735   8143   8159   8183*  8214   8220
$TN   = 000001          6057#  6061#
$TPB    001152          6319#  7746*
$TPFLG  001157          6323#  7679
$TPS    001150          6318#  7369   7370   7392   7397   7744
$TRAP   014052          6240   6543   8381#
$TRAP2  014074          8392#  8403
$TRPAD  014106          8386   8403#
$TSTM   001004          6283#
$TSTNM  001102          6296#  6651*  7471*  7977   8013   8099   8121*  8122   8127
$TTYIN  013442          8243   8244   8261   8265#
$TYPDS  011644          7903#  8408
$TYPE   010614          7679#  7788   8404
$TYPEC  011026          7708   7715   7722   7727#  8185
$TYPEX  011146          7750   7752   7755#
$TYPOC  011442          7844#  8405
$TYPON  011456          7843   7846#  8407
$TYPOS  011416          7839#  8406
$UNIT   001206          6341#  6631*  6640   6641   7457*
$UNITM  001010          6285#
$USWR   001220          6348#
$VECT1  001244          6373#  6580
$VECT2  001246          6374#
$XOFF = 000023          6056#  7732
$XON  = 000021          6055#  7739   8227
$XTSTR  012446          8084#
$OFILL  011641          7840*  7844*  7854   7889#
.     = 015366          6242#  6246#  6251#  6258   6259#  6261#  6263#  6264#  6270   6271#  6273#  6275#  6293#
                        6536   6550   6551   6604#  7809#  7957#  8063#  8265#  8341   8365   8544#
.$X   = 001000          6270#  6275

. ABS.  015366     000

 ERRORS DETECTED:  0

 CVDRAC,CVDRAC/CRF:SYM/NL:TOC=SYSMAC.SML,CVDRAC.P11
 RUN-TIME: 9 12 .8 SECONDS
 RUN-TIME RATIO: 77/22=3.3
 CORE USED:  36K  (72 PAGES)
```