DRV11B

DRV11B DMA INTFC DIAG AH-T086B-MC
CVDRABO

FICHE 1 OF 1

digital

```
                                .REM %
```

IDENTIFICATION

PRODUCT CODE: AC-8178B-MC

PRODUCT NAME: CVDRAB0 DRV11B DMA INTFC DIAG

DATE: AUGUST 1981

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1.0 ABSTRACT

2.0 REQUIREMENTS
2.1 EQUIPMENT
2.2 STORAGE

3.0 LOADING PROCEDURE

4.0 STARTING PROCEDURE
4.1 PROGRAM START

5.0 SOFTWARE SWITCH REGISTER
5.1 OPTIONS
5.2 CONTROL

6.0 ERROR REPORTING
6.1 ERROR COMMENT
6.2 ERROR DATA

7.0 MISCELLANEOUS
7.1 DRV11B BUS & VECTOR ADDRESS MODIFICATION
7.2 XXDP/APT NOTES
7.3 POWER FAIL
7.4 MULTIPLE DRV11B INTERFACE TESTING
7.5 RESTRICTIONS

8.0 EXECUTION TIME

9.0 PROGRAM TEST DESCRIPTIONS
9.1 GENERAL
9.2 REGISTER TESTS
9.3 BYTE ADDRESSING TESTS
9.4 'FNCT' TO 'STAT' WRAP AROUND TEST
9.5 READY INTERRUPT TEST
9.6 NPR DATA TRANSFER TESTS
9.7 MAINT MODE NPR DATA TRANSFER TESTS
9.8 BURST & NON-BURST MODE TESTS
9.9 'NEX' ERROR CONDITION TEST

10.0 LISTING

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

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

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

                                2.1     EQUIPMENT

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

                                2.2     STORAGE

                                        THE PROGRAM USES THE LOWER 4K OF MEMORY.

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

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

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

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

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

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

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

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

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

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

5.2 CONTROL

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

6.0 ERROR REPORTING

6.1 ERROR COMMENT

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

6.2 ERROR DATA

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

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

                7.1     DRV11B BUS & VECTOR ADDRESS MODIFICATION

                        MODIFY LOCATION '$BASE' IF BASE BUS ADDRESS IS NOT 172410.
                        MODIFY LOCATION '$VECT1' IF VECTOR ADDRESS IS NOT 124.

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

                7.2     XXDP/APT NOTES

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

                        IF THE PROCESSOR USED IS KDF11-B:
                             FOR APT SET $SWREG BIT07 TO 1 (000200)
                             FOR XXDP CHAINS SET LOCATION 176 (SWR) BIT07 TO 1
                                TO DO THIS UNDER XXDP:
                                  1. R UPD2
                                  2. LOAD VDRAB?.BIC
                                  3. MOD 176
                                     THE TERMINAL WILL RESPOND: 176/000000
                                  4. NOW TYPE IN 200
                                  5. BE CAREFUL: AT THIS POINT IT IS NECESARY TO DELETE THE FILE
                                     FROM THE DISK.
                                     DEL DL0:VDRAB?.BIC (IF MEDIA IS RL ON DRIVE 0)
                                6. DUMP VDRAB0.BIC
                                   TO BE SAFE, SKIP 5. AND DUMP THE FILE UNDER DIFFERENT NAME
                                   WITH .BIC EXTENSION.
                           NOTE: THIS PROCEDURE ASSUMES THAT DIAGNOSTIC IS ON THE DISK
                                 FROM WHICH THE SYSTEM IS BOOTED. IF THIS IS UNTRUE
                                 IN 2. AND 6. THE OPERATOR HAVE TO SPECIFY THE DRIVE BEFORE
                                 THE NAME.

                7.3     POWER FAIL

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

                7.4     MULTIPLE DRV11B INTERFACE TESTING

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

                7.5     RESTRICTIONS

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

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


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


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


                            9.1     GENERAL

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

                            9.2     REGISTER TESTS

                                    THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:

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

                            9.3     BYTE ADDRESSING TESTS

                                    1.  COMMAND/STATUS
                                    2.  DATA BUFFER

                            9.4     'FNCT' TO 'STAT' WRAP AROUND TEST

                            9.5     READY INTERRUPT TEST

                            9.6     NPR DATA TRANSFER TESTS

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

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

```
                                9.7     MAINT MODE NPR DATA TRANSFER TESTS

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

                                9.8     BURST & NON-BURST MODE TESTS

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

                                9.9     'NEX' ERROR CONDITION TEST


                                10.0    LISTING
                                        -------

                                %
```

```
                                .TITLE  MAINDEC-11-CVDRA-B DRV11B DMA INTERFACE DIAGNOSTIC
(1)                             ;*COPYRIGHT (C) 1981
(1)                             ;*DIGITAL EQUIPMENT CORP.
(1)                             ;*MAYNARD, MASS. 01754
(1)                             ;*
(1)                             ;*PROGRAM BY R. MOORE
(1)                             ;*
(1)                             ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)                             ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
(1)                             ;*
(1)     000001                  $TN=1
(1)     160000                  $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
372     167400                  $SWR=167400
373     000300                  $SWRMK=300
374     000001                  $TN=1
375                             .SBTTL  OPERATIONAL SWITCH SETTINGS
(1)                             ;*
(1)                             ;*      SWITCH                  USE
(1)                             ;*      ------          --------------------
(1)                             ;*        15            HALT ON ERROR
(1)                             ;*        14            LOOP ON TEST
(1)                             ;*        13            INHIBIT ERROR TYPEOUTS
(1)                             ;*        11            INHIBIT ITERATIONS
(1)                             ;*        10            BELL ON ERROR
(1)                             ;*         9            LOOP ON ERROR
(1)                             ;*         8            LOOP ON TEST IN SWR<5:0>
376                             .SBTTL  BASIC DEFINITIONS
(1)
(1)                             ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)     001100                  STACK=  1100
(1)                             .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
(1)                             .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)                             ;*MISCELLANEOUS DEFINITIONS
(1)     000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
(1)     000012                  LF=     12              ;;CODE FOR LINE FEED
(1)     000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
(1)     000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)     177776                  PS=     177776          ;;PROCESSOR STATUS WORD
(1)                             .EQUIV  PS,PSW
(1)     177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
(1)     177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)     177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
(1)     177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
(1)
(1)                             ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)     000000                  R0=     %0              ;;GENERAL REGISTER
(1)     000001                  R1=     %1              ;;GENERAL REGISTER
(1)     000002                  R2=     %2              ;;GENERAL REGISTER
(1)     000003                  R3=     %3              ;;GENERAL REGISTER
(1)     000004                  R4=     %4              ;;GENERAL REGISTER
(1)     000005                  R5=     %5              ;;GENERAL REGISTER
(1)     000006                  R6=     %6              ;;GENERAL REGISTER
(1)     000007                  R7=     %7              ;;GENERAL REGISTER
(1)     000006                  SP=     %6              ;;STACK POINTER
(1)     000007                  PC=     %7              ;;PROGRAM COUNTER
```

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

```
(1)                                 .EQUIV  BIT09,BIT9
(1)                                 .EQUIV  BIT08,BIT8
(1)                                 .EQUIV  BIT07,BIT7
(1)                                 .EQUIV  BIT06,BIT6
(1)                                 .EQUIV  BIT05,BIT5
(1)                                 .EQUIV  BIT04,BIT4
(1)                                 .EQUIV  BIT03,BIT3
(1)                                 .EQUIV  BIT02,BIT2
(1)                                 .EQUIV  BIT01,BIT1
(1)                                 .EQUIV  BIT00,BIT0
(1)
(1)                                 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)           000004                ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)           000010                RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)           000014                TBITVEC=14              ;;"T" BIT
(1)           000014                TRTVEC= 14              ;;TRACE TRAP
(1)           000014                BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)           000020                IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)           000024                PWRVEC= 24              ;;POWER FAIL
(1)           000030                EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)           000034                TRAPVEC=34              ;;"TRAP" TRAP
(1)           000060                TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)           000064                TPVEC=  64              ;;TTY PRINTER VECTOR
(1)           000240                PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
377           172410                ABASE=  172410          ;BASE DRV11B BUS ADRS EQUATE
378           000124                AVECT1= 000124          ;BASE DRV11B VECTOR ADRS EQUATE -
379           000001                ADEVM=  1               ;DEFAULT TO ONE DRV11B
380           106427                MTPS=106427             ;INSTR EQUATE THAT MOVES BYTE TO PSW
381  000000   177522                TMAIN: 177522           ;MAINTENCE REGISTER(FOR USE WITH KDF11-B)
382                                 .SBTTL  TRAP CATCHER
(1)
(1)           000000                        .=0
(1)                                 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)                                 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)                                 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)           000174                        .=174
(1)  000174   000000                DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)  000176   000000                SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
(1)                                 .SBTTL  STARTING ADDRESS(ES)
(1)  000200   000137  001550                JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
383           000100                .=100
384  000100   000104  000200  000002       .WORD   104,200,2       ;IF 'B EVENT' ON Q BUS IS CONNECTED
385                                                               ;IGNORE IT'S INTERRUPT - JUST DO A RTI
```

```
                                .SBTTL  ACT11 HOOKS

                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
        000106                          $SVPC=.                 ;SAVE PC
        000046                          .=46
000046  010056                          $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
001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114      000                 $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115      001                 $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134      000                 $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135      000                 $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154      000                 $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155      002                 $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156      012                 $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157      000                 $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001162  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001164  177607  000377        $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001170      077                 $QUES:  .ASCII  /?/             ;;QUESTION MARK
001171      015                 $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001172  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;;*****************************************************************************
                                        .SBTTL  APT MAILBOX-ETABLE

                                        ;;*****************************************************************************
                                        .EVEN
001174                          $MAIL:                          ;;APT MAILBOX
001174  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001176  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001200  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001202  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
001204  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001206  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001210  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
```

```
    (2)  001212  000000              $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
    (2)  001214                      $ETABLE:                ;;APT ENVIRONMENT TABLE
    (2)  001214     000              $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
    (2)  001215     000              $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
    (2)  001216  000000              $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
    (2)  001220  000000              $USWR:  .WORD   AUSWR   ;;USER SWITCHES
    (2)  001222  000000              $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
    (2)                              ;*                      BITS 15-11=CPU TYPE
    (2)                              ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
    (2)                              ;*                              11/70=06,PDQ=07,Q=10
    (2)                              ;*                      BIT 10=REAL TIME CLOCK
    (2)                              ;*                      BIT  9=FLOATING POINT PROCESSOR
    (2)                              ;*                      BIT  8=MEMORY MANAGEMENT
    (2)  001224     000              $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
    (2)  001225     000              $MTYP1: .BYTE   AMTYP1  ;;MEM. TYP.,BLK#1
    (2)                              ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
    (2)                              ;*                              900 NSEC CORE=001
    (2)                              ;*                              300 NSEC BIPOLAR=002
    (2)                              ;*                              500 NSEC MOS=003
    (2)  001226  000000              $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
    (2)                              ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
    (2)  001230     000              $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
    (2)  001231     000              $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
    (2)  001232  000000              $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
    (2)  001234     000              $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
    (2)  001235     000              $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
    (2)  001236  000000              $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
    (2)  001240     000              $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
    (2)  001241     000              $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
    (2)  001242  000000              $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
    (2)  001244  000124              $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
    (2)  001246  000000              $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
    (2)  001250  172410              $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
    (2)  001252  000001              $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
    (2)  001254  000000              $CDW1:  .WORD   ACDW1   ;;CONTROLLER DES     'ON WORD#1
    (2)  001256  000000              $CDW2:  .WORD   ACDW2   ;;CONTROLLER DE      ION WORD#2
    (2)  001260  000000              $DDW0:  .WORD   ADDW0   ;;DEVICE DESCR'      ORD#0
    (2)  001262  000000              $DDW1:  .WORD   ADDW1   ;;DEVICE DESC'       WORD#1
    (2)  001264  000000              $DDW2:  .WORD   ADDW2   ;;DEVICE DESC..  JR WORD#2
    (2)  001266  000000              $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
    (2)  001270  000000              $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
    (2)  001272  000000              $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
    (2)  001274  000000              $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
    (2)  001276  000000              $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
    (2)  001300  000000              $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
    (2)  001302  000000              $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
    (2)  001304  000000              $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
    (2)  001306  000000              $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
    (2)  001310  000000              $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
    (2)  001312  000000              $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
    (2)  001314  000000              $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
    (2)  001316  000000              $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
    (2)
    (2)
    (2)  001320                      $ETEND:
    (2)
```

```
(1)                                 .SBTTL  ERROR POINTER TABLE
(1)
(1)                                 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1)                                 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1)                                 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1)                                 ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1)                                 ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1)                                 ;*      EM              ;;POINTS TO THE ERROR MESSAGE
(1)                                 ;*      DH              ;;POINTS TO THE DATA HEADER
(1)                                 ;*      DT              ;;POINTS TO THE DATA
(1)                                 ;*      DF              ;;POINTS TO THE DATA FORMAT
(1)
(1)
(1)  001320                         $ERRTB:
391                                 ;ERROR  1
392  001320  014150                         EM1             ;REG TIMEOUT ER
393  001322  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
394  001324  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
395  001326  000000                         0
396
397                                 ;ERROR  2
398  001330  014167                         EM2             ;REG READ/WRITE ER
399  001332  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
400  001334  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
401  001336  000000                         0
402
403                                 ;ERROR  3
404  001340  014211                         EM3             ;BUS RESET ER
405  001342  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
406  001344  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
407  001346  000000                         0
408
409                                 ;ERROR  4
410  001350  014226                         EM4             ;FNCT BITS FAILED TO SET STAT BITS
411  001352  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
412  001354  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
413  001356  000000                         0
414
415                                 ;ERROR  5
416  001360  014270                         EM5             ;READY INTR FAILURE
417  001362  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
418  001364  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
419  001366  000000                         0
420
421                                 ;ERROR  6
422  001370  014313                         EM6             ;READY CLR OR SET ER
423  001372  015041                         DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
424  001374  015212                         DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
425  001376  000000                         0
```

```
                                ;ERROR  7
001400  014337                          EM7             ;STATUS ER ON XFER
001402  015041                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001404  015212                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001406  000000                          0

                                ;ERROR  10
001410  014361                          EM10            ;WORD COUNT ER ON XFER
001412  015041                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001414  015212                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001416  000000                          0

                                ;ERROR  11
001420  014407                          EM11            ;BUFFER ADRS ER ON XFER
001422  015041                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001424  015212                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001426  000000                          0

                                ;ERROR  12
001430  014436                          EM12            ;DATA ER FROM MEM
001432  015106                          DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001434  015226                          DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001436  000000                          0

                                ;ERROR  13
001440  014457                          EM13            ;DATA ER TO MEM
001442  015106                          DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001444  015226                          DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001446  000000                          0

                                ;ERROR  14
001450  014476                          EM14            ;SINGLE CYCLE OFF DID NOT LOCK OUT CPU
001452  015163                          DH3             ;ERRPC  TSTNUM  BUSADR
001454  015244                          DT3             ;$ERRPC TSTNUM  $BDADR
001456  000000                          0

                                ;ERROR  15
001460  014544                          EM15            ;SINGLE CYCLE ON LOCKED OUT CPU
001462  015163                          DH3             ;ERRPC  TSTNUM  BUSADR
001464  015244                          DT3             ;$ERRPC TSTNUM  $BDADR
001466  000000                          0

                                ;ERROR  16
001470  014732                          EM16            ;NEX LOGIC ER
001472  015041                          DH1             ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001474  015212                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001476  000000                          0

                                ;ERROR  17
001500  014747                          EM17            ;CYCLE FAILED TO CLK DBR (IN)
001502  015041                          DH1             ;ERRPC  TSTNUM  BDADR   GDDAT   BDDAT
001504  015212                          DT1             ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001506  000000                          0
```

```
                                ;ERROR  20
001510  015004                          EM20            ;DATA ER FROM I/O PAGE (XCSR)
001512  015106                          DH2             ;ERRPC  TSTNUM  BUSADR  ADRS    EXPCT   RCVD
001514  015226                          DT2             ;$ERRPC TSTNUM  $BDADR  $GDADR  $GDDAT  $BDDAT
001516  000000                          0

                                ;DRV11B BUS REGISTER ADDRESS POINTERS

001520  172410                  DRVWCR: 172410          ;WORD COUNT
001522  172412                  DRVBAR: 172412          ;BUFFER ADDRESS
001524  172414                  DRVCSR: 172414          ;COMMAND/STATUS
001526  172416                  DRVDBR: 172416          ;DATA BUFFER

                                ;DRV11B VECTOR ADDRESS POINTERS

001530  000124                  DRVCT0: 124             ;READY, NEX & INCOMPLETE DATIO VECTOR
001532  000126                  DRVCT2: 126             ;NEW PSW ON INTR

                                ;COMMON PROGRAM LOCATION(S)

001534  000000                  TSTNUM: 0               ;CONTAINS TEST NUMBER ON ERROR
001536  000001                  DMAP:   1               ;DEVICE MAP - EA BIT SAYS TEST THAT DRV11B
001540  000000                  CORSZ:  0               ;CONTAINS 1ST NON-EXISTANT MEM ADRS
001542  015400                  DBUFP:  DBUF            ;CONTAINS CURRENT 4K NPR BUFFER ADRS
001544  000000                  IOPAGE: 0               ;CONTAINS ADDRESS FOR I/O TRANSFERS
001546  000000                  KDF:    0               ;IF KDF11-B
```

```
                                        .SBTTL  PROGRAM START
001550                                  START:
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001550  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001554  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001556  022706  001140                  CMP     #SWR,R6 ;;DONE?
001562  001374                          BNE     .-6             ;;LOOP BACK IF NO
001564  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001570  012737  012446  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001576  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001604  012737  012104  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001612  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001620  012737  014066  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001626  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001634  012737  013662  000024          MOV     #$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     #DRVCT0,R0      ;SET UP DRV11B VECTOR ADRS POINTER
002042  013701  001244          MOV     $VECT1,R1       ;GET BASE VECTOR ADRS
002046  042701  170000          BIC     #170000,R1      ;CLR OUT PRIORITY BITS
002052  010120          SETUP3: MOV     R1,(R0)+        ;
002054  062701  000002          ADD     #2,R1           ;POINT TO NEXT
002060  022700  001534          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$:          .ASCIZ  <CRLF>#MD-11-CVDRA-B  DRV11B DMA INTERFACE DIAG  #<CRLF>
64$:
                        INC     #-1             ;FIRST PASS?
                        BNE     100$            ;BRANCH IF NO
                        CMPB    #1,$AUTOB       ;MANUAL INTERVENTION PERMITTED?
                        BEQ     100$            ;BR IF NO
                        TYPE,   KDF11B          ;ASK ABOUT KDF11-B
                        RDCHR                   ;READ ANSWER
                        MOV     (SP)+,KDF       ;STORE ANSWER
                        TYPE,   KDF             ;ECHO ANSWER
                        CMPB    KDF,#131        ;IS IT KDF11-B?
                        BNE     100$            ;BRANCH IF NOT
                        BISB    #BIT07,@SWR     ;IF YES,SET SWREG
100$:           TST     CORSZ           ;TEST IF FIRST PASS
                        BNE     CORSZR          ;BR IF NOT
                        TYPE,   WARN            ;TELL THE OPERATOR TO TURN OFF DMA REFRESH
                ;;*****************************************************************
                ;LET'S SEE HOW MUCH MEM WE HAVE
                ;;*****************************************************************
CORSZR: MOV     #20000,R0       ;USE R0 TO LOOK
                        MOV     #2$,@#ERRVEC    ;SET UP TIME OUT RETURN ADRS
1$:             TST     (R0)+           ;TAKE A LOOK
                        BR      1$              ;UNTIL TIMEOUT
2$:             BIC     #17777,R0       ;POINT TO 1ST NON-EXSISTANT 4K BLK
                        MOV     R0,CORSZ        ;SAVE FOR LATER
                        MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE VECTOR
                        MOV     #DBUF,DBUFP     ;INITIALIZE TO LOWEST 4K
                        MOV     #0,$UNIT        ;SET UP UNIT COUNT
                        MOV     $DEVM,DMAP      ;GET THE # & POSITION OF DRV11B'S
                        BIC     #177400,DMAP    ;UP TO 8 ONLY
                        BEQ     RESTRT          ;GO CONTINUE AS IF SOMETHING WAS SELECTED
                        BIT     #1,DMAP         ;IS 1ST DRV11B SELECTED?
                        BNE     RESTRT          ;BR IF SO
                        JMP     NXDEV1          ;NO - GO ADVANCE BASE DRV11B ADDRESSES
RESTR1: MTPS    #200            ;SET PRIORITY TO HIGHEST LEVEL
                        MOV     #STACK,SP       ;ALWAYS RESET STACK PTR
                        MOV     $UNIT,$DEVCT    ;LOAD APT COUNTER
                        MOV     $UNIT,R0        ;MAKE AN INDEX
```

```
                                        ASL     R0              ; VALUE
                                        MOV     DRVWCR,$DDW0(R0)        ;SAVE THE BUS ADDRESS
                                        RESET                   ;INITIALIZE DRV11B BEFORE TESTING
                                ;;**************************************************************
                                ;*TEST 1         TEST THAT ALL DRV11B REGS ARE ACCESSIBLE
                                ;;**************************************************************
                                TST1:   <NOP>
                                        MOV     #10$,$LPADR     ;;SET SCOPE LOOP ADDRESS
                                        MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
                                10$:    MOVB    #1,$TSTNM       ;SET TO TEST #1
                                        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                        CLR     $GDDAT          ;NO DATA COMPARE
                                        CLR     $BDDAT          ;NO DATA COMPARE
                                        MOV     #2$,@#ERRVEC    ;SET UP TIMEOUT RETURN ADDRS
                                        MOV     DRVWCR,R0       ;SET UP 1ST DRV11 BUS ADRS
                                        MOV     #4,R1           ;SET UP REG COUNT
                                1$:     MOV     R0,$BDADR       ;SET UP CURRENT DRV BUS ADRS
                                        TST     (R0)            ;SEE IF THERE
                                        TST     (R0)+           ;BUMP TO NEXT
                                        DEC     R1              ;COUNT 4 OF THEM
                                        BEQ     3$              ;BR IF ALL DONE
                                        BR      1$              ;TRY NEXT
                                2$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
                                        ERROR   1               ;BUS ADRS INDICATED DID NOT RESPOND
                                3$:     MOV     #ERRVEC+2,@#ERRVEC  ;RESTORE LOC 4

                                ;;**************************************************************
                                ;*TEST 2         TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
                                ;;**************************************************************
                                TST2:   SCOPE
                                        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                        MOV     DRVWCR,$BDADR   ;SET UP WC REG ADRS
                                        CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
                                        MOV     #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
                                1$:     MOV     $GDDAT,@DRVWCR  ;LD WC
                                        MOV     @DRVWCR,$BDDAT  ;READ IT BACK
                                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                                        BEQ     2$              ;BR IF SO
                                        ERROR   2               ;WORD COUNT WRITE/READ FAILURE
                                2$:     COM     $GDDAT          ;COMPELEMENT ZERO
                                        COM     R0              ;R0 SAYS SHIFT LEFT WHEN = 0
                                        BNE     1$              ;TRY THE COMPLEMENT IF R0 NOT 0
                                        ASL     $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
                                        INC     $GDDAT          ;KEEP LSB SET
                                        BCS     1$              ;AGAIN TILL ALL PATRNS DONE

                                ;;**************************************************************
                                ;*TEST 3         TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
                                ;;**************************************************************
                                TST3:   SCOPE
                                        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                        MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
                                        CLR     R0              ;R0 SAYS SHIFT PTRN WHEN 0
                                        MOV     #-4,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
                                1$:     MOV     $GDDAT,@DRVBAR  ;LD BA
                                        MOV     @DRVBAR,$BDDAT  ;READ IT BACK
```

```
002712  042737  000001  001126          BIC     #BIT00,$BDDAT     ;DON'T WANT BIT00
002720  023737  001124  001126          CMP     $GDDAT,$BDDAT     ;CORRECT?
002726  001401                          BEQ     2$                ;BR IF SO
002730  104002                          ERROR   2                 ;BUS ADRS WRITE/READ FAILURE
002732  005137  001124          2$:     COM     $GDDAT            ;COMPLEMENT ZERO
002736  042737  000001  001124          BIC     #BIT00,$GDDAT     ;BIT 00 NOT INVOLVED
002744  005100                          COM     R0                ;R0 SAYS SHIFT LEFT WHEN = 0
002746  001353                          BNE     1$                ;TRY THE COMPLEMENT IF R0 NOT 0
002750  006337  001124                  ASL     $GDDAT            ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
002754  103004                          BCC     TST4              ;;NEXT TEST IF BIT 15 DONE
002756  062737  000002  001124          ADD     #2,$GDDAT         ;KEEP ADDR LSB SET
002764  000744                          BR      1$                ;AGAIN TILL ALL PATTERNS DONE


                                ;;*****************************************************************
                                ;*TEST 4        TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
                                ;;*****************************************************************
002766  000004                  TST4:   SCOPE
002770  012737  003014  001110          MOV     #1$,$LPERR        ;SET UP SCOPE LOOP ADRS
002776  013737  001526  001122          MOV     DRVDBR,$BDADR     ;SET UP DB REG ADRS
003004  005000                          CLR     R0                ;R0 SAYS SHIFT PTRN WHEN 0
003006  012737  177776  001124          MOV     #-2,$GDDAT        ;FLOAT 0 RIGHT TO LEFT
003014  013777  001124  176504  1$:     MOV     $GDDAT,@DRVDBR    ;LD DB
003022  017737  176500  001126          MOV     @DRVDBR,$BDDAT    ;READ IT BACK
003030  023737  001124  001126          CMP     $GDDAT,$BDDAT     ;CORRECT?
003036  001401                          BEQ     2$                ;BR IF SO
003040  104002                          ERROR   2                 ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
003042  005137  001124          2$:     COM     $GDDAT            ;COMPLEMENT ZERO
003046  005100                          COM     R0                ;R0 SAYS SHIFT LEFT WHEN = 0
003050  001361                          BNE     1$                ;TRY THE COMPLEMENT IF R0 NOT 0
003052  006337  001124                  ASL     $GDDAT            ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
003056  005237  001124                  INC     $GDDAT            ;KEEP LSB SET
003062  103754                          BCS     1$                ;AGAIN TILL ALL PATRNS DONE


                                ;;*****************************************************************
                                ;*TEST 5        TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE
                                ;;*****************************************************************
003064  000004                  TST5:   SCOPE
003066  016700  001526                  MOV     DRVDBR,R0         ;GET DB REG ADRS
003072  010037  001122                  MOV     R0,$BDADR         ;SET UP DB REG ADRS
003076  005010                          CLR     (R0)              ;ZERO DATA BUFFER REG
003100  012737  177177  001124          MOV     #177177,$GDDAT    ;LD EXPECTED
003106  012737  077776  001126          MOV     #77776,$BDDAT     ;SEND DATA FROM '$BDDAT'
003114  153760  001126  000001          BISB    $BDDAT,1(R0)      ;LOAD HI BYTE DB
003122  153710  001127                  BISB    $BDDAT+1,(R0)     ;LOAD LO BYTE DB
003126  011037  001126                  MOV     (R0),$BDDAT       ;READ IT BACK
003132  023737  001124  001126          CMP     $GDDAT,$BDDAT     ;CORRECT?
003140  001401                          BEQ     TST6              ;;NEXT TEST IF SO
003142  104002                          ERROR   2                 ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG


                                ;;*****************************************************************
                                ;*TEST 6        TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
                                ;;*****************************************************************
003144  000004                  TST6:   SCOPE
003146  012737  000010  001160          MOV     #10,$TIMES        ;;DO 10 ITERATIONS
003154  005037  001124                  CLR     $GDDAT            ;LD EXPECTED
003160  012777  177777  176332          MOV     #-1,@DRVWCR       ;SET ALL BITS - WC REG
```

```
   656  003166  012777  177776  176326          MOV     #-2,@DRVBAR     ;SET ALL BITS - BUS ADRS REG
   657  003174  012777  177777  176324          MOV     #-1,@DRVDBR     ;SET ALL BITS - DB OUT REG
   658  003202  000005                          RESET                   ;DO A BUS RESET
   659  003204  017737  176310  001126          MOV     @DRVWCR,$BDDAT  ;READ WC REG
   660  003212  001404                          BEQ     1$              ;BR IF CLRED
   661  003214  013737  001520  001122          MOV     DRVWCR,$BDADR   ;SET UP WC REG ADRS
   662  003222  104003                          ERROR   3               ;RESET FAILED TO CLR WC REG
   663  003224  017737  176272  001126  1$:     MOV     @DRVBAR,$BDDAT  ;READ BUS ADRS REG
   664  003232  042737  000001  001126          BIC     #BIT00,$BDDAT   ;DON'T WANT BIT00
   665  003240  001404                          BEQ     2$              ;BR IF CLRED
   666  003242  013737  001522  001122          MOV     DRVBAR,$BDADR   ;SET UP BA REG ADRS
   667  003250  104003                          ERROR   3               ;RESET FAILED TO CLR BUS ADRS REG
   668  003252  017737  176250  001126  2$:     MOV     @DRVDBR,$BDDAT  ;READ DATA BUFFER REG
   669  003260  001404                          BEQ     TST7            ;;NEXT TEST IF CLRED
   670  003262  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DB REG ADRS
   671  003270  104003                          ERROR   3               ;RESET FAILED TO CLR DATA BUFFER OUT REG
   672
   673                                  ;;*********************************************************************
   (3)                                  ;*TEST 7        TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
   (3)                                  ;;*********************************************************************
   (2)  003272  000004          TST7:   SCOPE
   (1)  003274  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   674  003302  106427  000200                  MTPS    #200            ;DON'T WANT ANY INTRS
   675  003306  004537  010112                  JSR     R5,SETVEC       ;SET UP INTR RETURN ADRS IN CASE
   676  003312  003410                          3$                      ;RETURN TO 3$ ON ILLEGAL INTR
   677  003314  013737  003334  001110          MOV     1$,$LPERR       ;SET UP SCOPE LOOP ADRS
   678  003322  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
   679  003330  012700  160000                  MOV     #160000,R0      ;START AT 0 - HI BITS FOR NOISE
   680  003334  010037  001124          1$:     MOV     R0,$GDDAT       ;LD EXPECTED
   681  003340  042737  167201  001124          BIC     #167201,$GDDAT  ;MASK TO WRITEABLE BITS
   682  003346  010077  176152                  MOV     R0,@DRVCSR      ;LD CSR
   683  003352  017737  176146  001126          MOV     @DRVCSR,$BDDAT  ;READ IT BACK
   684  003360  042737  007200  001126          BIC     #7200,$BDDAT    ;DON'T LOOK AT STAT & RDY BITS
   685  003366  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
   686  003374  001401                          BEQ     2$              ;BR IF SO
   687  003376  104002                          ERROR   2               ;CONTROL/STATUS REG WRITE/READ FAILURE
   688  003400  062700  000002          2$:     ADD     #2,R0           ;ADVANCE COUNT PATTERN
   689  003404  001353                          BNE     1$              ;WRITE NEXT PATTERN IF NOT ALL TESTED
   690  003406  000413                          BR      4$              ;GO RESTORE VECTOR
   691  003410  022626                  3$:     CMP     (SP)+,(SP)+     ;FIX STACK - SHOULD NOT HAVE INT'ED
   692  003412  052737  000200  001124          BIS     #200,$GDDAT     ;CORRECT EXPECTED
   693  003420  017737  176100  001126          MOV     @DRVCSR,$BDDAT  ;READ CSR
   694  003426  042737  007000  001126          BIC     #7000,$BDDAT    ;DON'T WANT 'STAT' BITS
   695  003434  104005                          ERROR   5               ;CPU FAILED TO LOCK OUT DRV11B INTR REQ
   696  003436  004737  010132          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
   697
   698                                  ;;*********************************************************************
   (3)                                  ;*TEST 10       TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
   (3)                                  ;;*********************************************************************
   (2)  003442  000004          TST10:  SCOPE
   (1)  003444  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   699  003452  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
   700  003460  012737  000200  001124          MOV     #200,$GDDAT     ;LD EXPECTED
   701  003466  012777  177776  176030          MOV     #-2,@DRVCSR     ;LD ALL CSR BITS
   702  003474  000005                          RESET                   ;DO A BUS RESET
   703  003476  017737  176022  001126          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' - ISE 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
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        MTPS    #200            ;DONT WANT INTR YET
                                        RESET                   ;SET THE READY FLAG BY INIT
                                        MOV     #1$,@DRVCTO     ;SET UP PREMATURE INTR RETURN ADRS
                                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                        MOV     #300,$GDDAT     ;LD EXPECTED (READY + IE)
                                        MTPS    #0              ;ALLOW AN INTR
                                        CMP     (SP),(SP)       ;STALL
                                        MOV     #2$,@DRVCTO     ;SET UP EXPECTED INTR RETURN ADRS
                                        BIS     #BIT6,@DRVCSR   ;ENABLE THE EXPECTED INTERRUPT
                                        CMP     (SP),(SP)       ;STALL
                                        MOV     @DRVCSR,$BDDAT  ;GET THE CSR
                                        ERROR   5               ;READY FAILED TO CAUSE AN INTERRUPT
```

```
                                        BR      3$              ;GO RESTORE VECTOR
                                1$:     CMP     (SP)+,(SP)+     ;SHOULD NEVER GET HERE - IE NOT WORKING?
004772  017737  175522  001126        MOV     @DRVCSR,$BDDAT  ;GET THE CSR
004004  104005                        ERROR   5               ;READY INTERRUPTED WITHOUT THE IE BIT
004006  000411                        BR      3$              ;GO RESTORE VECTOR
004010  022626                        CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RETURN
004012  017737  175506  001126        MOV     @DRVCSR,$BDDAT  ;READ STATUS
004020  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
004026  001401                        BEQ     3$              ;BR IF SO
004030  104005                        ERROR   5               ;INCORRECT STATUS ON READY INTR
004032  004737  010132          3$:   JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                                ;;*******************************************************************
                                ;*TEST 14        TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
                                ;;*******************************************************************
004036  000004                TST14:  SCOPE
004040  106427  000200                MTPS    #200            ;DONT WANT ANY INTRS
004044  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
004052  005037  001124                CLR     $GDDAT          ;EXPECT 0
004056  012777  000001  175440        MOV     #1,@DRVCSR      ;SET GO WHICH SHOULD CLR READY
004064  017737  175434  001126        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
004072  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
004100  001401                        BEQ     1$              ;BR IF SO
004102  104006                        ERROR   6               ;THE GO BIT FAILED TO CLR READY
004104  052777  000004  175412  1$:   BIS     #4,@DRVCSR      ;FNCT 2 SHOULD SET READY
004112  012737  002204  001124        MOV     #2204,$GDDAT    ;LD EXPECTED
004120  017737  175400  001126        MOV     @DRVCSR,$BDDAT  ;GET CSR
004126  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
004134  001401                        BEQ     TST15           ;;NEXT TEST IF SET
004136  104006                        ERROR   6               ;FNCT 2 (VIA ATTN) FAILED TO SET READY

                                ;;*******************************************************************
                                ;*TEST 15        TEST THAT READY CONTROLS 'BAR' BIT00
                                ;;*******************************************************************
004140  000004                TST15:  SCOPE
004142  012777  000004  175354        MOV     #4,@DRVCSR      ;SET READY
004150  013737  001522  001122        MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
004156  012737  000001  001124        MOV     #1,$GDDAT       ;EXPECT LSB OF BAR
004164  005077  175332                CLR     @DRVBAR         ;CLR BAR
004170  017737  175326  001126        MOV     @DRVBAR,$BDDAT  ;READ BAR
004176  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
004204  001401                        BEQ     1$              ;BR IF SO
004206  104002                        ERROR   2               ;A00 FAILED TO READ A ONE(SB TIED TO RDY)
004210  012777  000001  175306  1$:   MOV     #1,@DRVCSR      ;SET GO(CLRS BAR BIT00)
004216  017737  175300  001126        MOV     @DRVBAR,$BDDAT  ;READ BAR
004224  001403                        BEQ     2$              ;BR IF ZERO
004226  005037  001124                CLR     $GDDAT          ;EXPECTED ZERO
004232  104002                        ERROR   2               ;WHEN RDY CLRED-A00 FAILED TO READ A ZERO
004234  012777  000004  175262  2$:   MOV     #4,@DRVCSR      ;INSURE RDY SET BEFORE ADVANCING

                                ;;*******************************************************************
                                ;*TEST 16        TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
                                ;;*******************************************************************
004242  000004                TST16:  SCOPE
004244  013737  001526  001122        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
004252  012737  125252  001124        MOV     #125252,$GDDAT  ;LD EXPECTED
```

```
                                                                M 2
MAINDEC-11-CVDRA-B DRV11B DMA INTERFACE DIAGNOSTIC      MACY11 30(1046)  11-SEP-81  12:22  PAGE 5-16
CVDRAB.P11    11-SEP-81 12:21           T16     TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)                SEQ 0025

   797  004260  013777  001124  175240          MOV     $GDDAT,@DRVDBR  ;LD DBR WITH #125252
   798  004266  012777  000400  175230          MOV     #400,@DRVCSR    ;SET CYCLE - SHOULD CLK DBR (IN)
   799  004274  012777  052525  175224          MOV     #52525,@DRVDBR  ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
   800  004302  017737  175220  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
   801  004310  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
   802  004316  001401                          BEQ     1$              ;BR IF SO
   803  004320  104017                          ERROR   17              ;CYCLE DID NOT LATCH DBR (IN) DATA
   804  004322  042777  000400  175174  1$:     BIC     #400,@DRVCSR    ;REMOVE CYCLE
   805  004330  012737  052525  001124          MOV     #52525,$GDDAT   ;NOW EXPECT #52525
   806  004336  017737  175164  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
   807  004344  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
   808  004352  001401                          BEQ     TST17           ;;NEXT TEST IF SO
   809  004354  104002                          ERROR   2               ;DBR FAILED TO READ WHEN CYCLE CLRED (NORMAL)
   810
   811                                  ;;*****************************************************************
   (3)                                  ;*TEST 17        TEST SINGLE "DATI" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
   (3)                                  ;;*****************************************************************
   (2)  004356  000004                  TST17:  SCOPE
   812  004360  012737  004402  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
   813  004366  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
   814  004372  004476                          2$                      ;RETURN TO 2$ ON INTR
   815  004374  012700  177776                  MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
   816  004400  005001                          CLR     R1              ;R1 CONTROLS DATA SHIFTING
   817  004402  012777  177777  175110  1$:     MOV     #-1,@DRVWCR     ;DO ONE XFER
   818  004410  012777  015400  175104          MOV     #DBUF,@DRVBAR   ;GET DATA WORD FROM "DBUF"
   819  004416  010037  015400                  MOV     R0,DBUF         ;SET UP MEM DATA
   820  004422  012777  000101  175074          MOV     #101,@DRVCSR    ;SET IE & GO
   821  004430  052777  000400  175066          BIS     #400,@DRVCSR    ;SET CYCLE
   822  004436  106427  000000                  MTPS    #0              ;ENABLE THE INTR
   823  004442  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
   824  004450  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
   825  004456  017737  175042  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
   826  004464  042777  000100  175032          BIC     #100,@DRVCSR    ;CLR IE
   827  004472  104005                          ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
   828  004474  000442                          BR      4$              ;GO RESTORE VECTOR
   829  004476  022626                  2$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
   830  004500  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
   831  004504  000700                          700                     ;CSR STATUS EXPECTED
   832  004506  000001                          1                       ;# OF XFERS
   833  004510  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
   834  004512  000433                          BR      4$              ;GO RESTORE VECTOR
   835  004514  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
   836  004516  000431                          BR      4$              ;GO RESTORE VECTOR
   837  004520  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
   838  004522  000427                          BR      4$              ;GO RESTORE VECTOR
   839  004524  012737  015400  001120          MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
   840  004532  010037  001124                  MOV     R0,$GDDAT       ;LD EXPECTED
   841  004536  017737  174764  001126          MOV     @DRVDBR,$BDDAT  ;READ DATA XFERED
   842  004544  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
   843  004552  001405                          BEQ     3$              ;BR IF SO
   844  004554  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
   845  004562  104012                          ERROR   12              ;DATA ER - DBR CONTAINS WRONG DATA
   846  004564  000406                          BR      4$              ;GO RESTORE VECTOR
   847  004566  005100                  3$:     COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
   848  004570  005101                          COM     R1              ;KEEP TRACK OF COMPLEMENT
   849  004572  001303                          BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
```

```
                                ASL     R0              ;WAS DONE - NOW SHIFT ZERO LEFT
                                INC     R0              ;KEEP LSB SET
                                BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
004602  004737  010132    4$:   JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                        ;;*******************************************************************
                        ;*TEST 20       TEST SINGLE "DATO" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
                        ;;*******************************************************************
004606  000004          TST20:  SCOPE
004610  012737  004632  001110        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
004616  004537  010112                JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
004622  004720                        2$                      ;RETURN TO 2$ ON INTR
004624  012700  177776                MOV     #-2,R0          ;FLOAT ZERO RIGHT TO LEFT
004630  005001                        CLR     R1              ;R1 CONTROLS DATA SHIFTINC
004632  012777  177777  174660  1$:   MOV     #-1,@DRVWCR     ;DO ONE XFER
004640  012777  015400  174654        MOV     #DBUF,@DRVBAR   ;WRITE DATA WORD TO "DBUF"
004646  010077  174654                MOV     R0,@DRVDBR      ;SET UP DATA IN DBR
004652  012777  000103  174644        MOV     #103,@DRVCSR    ;SET IE, GO & FNCT1 (C1 CONTROL)
004660  052777  000400  174636        BIS     #400,@DRVCSR    ;SET CYCLE
004666  106427  000000                MTPS    #0              ;ENABLE THE INTR
004672  013737  001524  001122        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
004700  012737  001702  001124        MOV     #1702,$GDDAT    ;LD EXPECTED
004706  017737  174612  001126        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
004714  104005                        ERROR   5               ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
004716  000442                        BR      4$              ;GO RESTORE VECTOR
004720  022626          2$:   CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
004722  004537  010156                JSR     R5,CKSTAT       ;GO CHECK STATUS
004726  001702                        1702                    ;CSR STATUS EXPECTED
004730  000001                        1                       ;# OF XFERS
004732  104007                        ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
                                                                ;CYCLE, READY, IE & FNCT 1
004734  000433                        BR      4$              ;GO RESTORE VECTOR
004736  104010                        ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
004740  000431                        BR      4$              ;GO RESTORE VECTOR
004742  104011                        ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
004744  000427                        BR      4$              ;GO RESTORE VECTOR
004746  012737  015400  001120        MOV     #DBUF,$GDADR    ;RETURN HERE IF OK - SET UP XFER ADRS
004754  010037  001124                MOV     R0,$GDDAT       ;LD EXPECTED
004760  013737  015400  001126        MOV     DBUF,$BDDAT     ;GET DATA XFERED
004766  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;CORRECT?
004774  001405                        BEQ     3$              ;BR IF SO
004776  013737  001526  001122        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
005004  104013                        ERROR   13              ;DATA ER - MEM CONTAINS WRONG DATA
005006  000406                        BR      4$              ;GO RESTORE VECTOR
005010  005100          3$:   COM     R0              ;RETURN HERE ON GOOD DATA - NOW COM PATRN
005012  005101                COM     R1              ;KEEP TRACK OF COMPLEMENT
005014  001306                BNE     1$              ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
005016  006300                ASL     R0              ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
005020  005200                INC     R0              ;KEEP LSB SET
005022  103703                BCS     1$              ;AGAIN TILL ZERO BIT IN CARRY
005024  004737  010132    4$:   JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                        ;;*******************************************************************
                        ;*TEST 21       TEST 200 "DATI" NPR TRANSFERS (BURST MODE)
                        ;;*******************************************************************
005030  000004          TST21:  SCOPE
```

```
(1)  005032  012737  000010  001160          MOV   #10,$TIMES      ;;DO 10 ITERATIONS
900  005040  004537  010112                  JSR   R5,SETVEC       ;GO SET UP INTERRUPT RETURN
901  005044  005142                          1$                    ;RETURN TO 1$ ON INTR
902  005046  004737  010336                  JSR   PC,LDBUF        ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
903  005052  012777  177470  174440          MOV   #-200.,@DRVWCR  ;LOAD WC REG - WILL DO 200 XFERS
904  005060  012777  015400  174434          MOV   #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
905  005066  106427  000000                  MTPS  #0              ;ENABLE THE INTR
906  005072  012777  000101  174424          MOV   #101,@DRVCSR    ;SET IE & GO
907  005100  052777  000400  174416          BIS   #400,@DRVCSR    ;SET CYCLE
908  005106  013737  001524  001122          MOV   DRVCSR,$BDADR   ;SET UP CSR ADRS
909  005114  012737  000700  001124          MOV   #700,$GDDAT     ;LD EXPECTED
910  005122  017737  174376  001126          MOV   @DRVCSR,$BDDAT  ;READ THE CSR
911  005130  042777  000100  174366          BIC   #100,@DRVCSR    ;CLR INTR ENABLE
912  005136  104005                          ERROR 5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
913  005140  000434                          BR    2$              ;GO RESTORE VECTOR
914  005142  022626                  1$:     CMP   (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
915  005144  004537  010156                  JSR   R5,CKSTAT       ;GO CHECK STATUS
916  005150  000700                          700                   ;CSR STATUS EXPECTED
917  005152  000310                          200.                  ;# OF XFERS
918  005154  104007                          ERROR 7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
919  005156  000425                          BR    2$              ;GO RESTORE VECTOR
920  005160  104010                          ERROR 10              ;RETURN HERE IF WC ER - EXPECTED 0
921  005162  000423                          BR    2$              ;GO RESTORE VECTOR
922  005164  104011                          ERROR 11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
923  005166  000421                          BR    2$              ;GO RESTORE VECTOR
924  005170  012737  016216  001120          MOV   #DBUF+620,$GDADR  ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
925  005176  012737  070707  001124          MOV   #70707,$GDDAT   ;LD EXPECTED
926  005204  017737  174316  001126          MOV   @DRVDBR,$BDDAT  ;DBR SHOULD HAVE LAST DATUM
927  005212  023737  001124  001126          CMP   $GDDAT,$BDDAT   ;CORRECT?
928  005220  001404                          BEQ   2$              ;BR IF SO
929  005222  013737  001526  001122          MOV   DRVDBR,$BDADR   ;SET UP DBR ADRS
930  005230  104012                          ERROR 12              ;DATA ER - DBR DID NOT CONTAIN EXPECTED LAST XFER
931  005232  004737  010132          2$:     JSR   PC,RSTVEC       ;GO RESTORE VECTOR
932
933                                  ;;*****************************************************************
(3)                                  ;*TEST 22       TEST 200 "DATO" NPR TRANSFERS (BURST MODE)
(3)                                  ;;*****************************************************************
(2)  005236  000004                  TST22:  SCOPE
(1)  005240  012737  000010  001160          MOV   #10,$TIMES      ;;DO 10 ITERATIONS
934  005246  004537  010112                  JSR   R5,SETVEC       ;GO SET UP INTERRUPT RETURN
935  005252  005352                          1$                    ;RETURN TO 1$ ON INTR
936  005254  012777  177470  174236          MOV   #-200.,@DRVWCR  ;WORD WC REG - WILL DO 200 XFER'S
937  005262  012777  015400  174232          MOV   #DBUF,@DRVBAR   ;SET UP CURRENT ADPS
938  005270  012777  177377  174230          MOV   #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
939  005276  106427  000000                  MTPS  #0              ;ENABLE THE INTR
940  005302  012777  000103  174214          MOV   #103,@DRVCSR    ;SET IE, FNCT 1 & GO
941  005310  052777  000400  174206          BIS   #400,@DRVCSR    ;SET CYCLE
942  005316  013737  001524  001122          MOV   DRVCSR,$BDADR   ;SET UP CSR ADRS
943  005324  012737  001702  001124          MOV   #1702,$GDDAT    ;LD EXPECTED
944  005332  017737  174166  001126          MOV   @DRVCSR,$BDDAT  ;READ THE CSR
945  005340  042777  000100  174156          BIC   #100,@DRVCSR    ;CLR INTR ENABLE
946  005346  104005                          ERROR 5               ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
947  005350  000416                          BR    2$              ;GO RESTORE VECTOR
948  005352  022626                  1$:     CMP   (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
949  005354  004537  010156                  JSR   R5,CKSTAT       ;GO CHECK STATUS
950  005360  001702                          1702                  ;CSR STATUS EXPECTED
```

```
   951  005362  000310                          200.                    ;# OF XFERS
   952  005364  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
   953                                                                  ;CYCLE, READY, IE & FNCT 1
   954  005366  000407                          BR      2$              ;GO RESTORE VECTOR
   955  005370  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
   956  005372  000405                          BR      2$              ;GO RESTORE VECTOR
   957  005374  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
   958  005376  000403                          BR      2$              ;GO RESTORE VECTOR
   959  005400  004737  010460                  JSR     PC,CKDAT        ;RETURN HERE IF OK - NOW GO CHECK DATA
   960  005404  104013                          ERROR   13              ;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
   961  005406  004737  010132          2$:     JSR     PC,RSTVEC       ;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
   962
   963                                ;;*********************************************************************
   (3)                                ;*TEST 23        TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
   (3)                                ;;*********************************************************************
   (2)  005412  000004                TST23:  SCOPE
   (1)  005414  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   964  005422  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
   965  005430  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
   966  005434  005544                          3$                      ;RETURN TO 3$ ON INTR
   967  005436  012700  000010                  MOV     #10,R0          ;DO EIGHT 200 WORD XFER'S
   968  005442  005037  001126                  CLR     $BDDAT          ;USR $BDDAT AS A COUNTER
   969  005446  012777  177470  174044  1$:     MOV     #-200.,@DRVWCR  ;DO 200 XFERS (DATI'S)
   970  005454  012777  015400  174040          MOV     #DBUF,@DRVBAR   ;FROM DBUF
   971  005462  106427  000000                  MTPS    #0              ;ALLOW AN INTR
   972  005466  012777  000101  174030          MOV     #101,@DRVCSR    ;SET IE & GO
   973  005474  052777  000400  174022          BIS     #400,@DRVCSR    ;SET CYCLE
   974  005502  000240                          NOP                     ;FREEBEE
   975  005504  000240                          NOP
   976  005506  000240                          NOP
   977  005510  005237  001126          2$:     INC     $BDDAT          ;START COUNTING - SHOULD NEVER GET HERE
   978  005514  001375                          BNE     2$              ;UNTIL 64K
   979  005516  012737  000700  001124          MOV     #700,$GDDAT     ;LD EXPECTED
   980  005524  017737  173774  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
   981  005532  042777  000100  173764          BIC     #100,@DRVCSR    ;CLR IE
   982  005540  104005                          ERROR   5               ;NO INTERRUPT ON 200 DATI'S
   983  005542  000407                          BR      4$              ;GO RESTORE VECTOR
   984  005544  022626                3$:       CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
   985  005546  005300                          DEC     R0              ;DONE 8 TIMES?
   986  005550  001336                          BNE     1$              ;BR IF NOT
   987  005552  005737  001126                  TST     $BDDAT          ;SHOULD STILL BE ZERO
   988  005556  001401                          BEQ     4$              ;BR IF SO
   989  005560  104014                          ERROR   14              ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
   990  005562  004737  010132          4$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
   991
   992                                ;;*********************************************************************
   (3)                                ;*TEST 24        TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
   (3)                                ;;*********************************************************************
   (2)  005566  000004                TST24:  SCOPE
   (1)  005570  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   993  005576  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
   994  005604  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
   995  005610  005716                          3$                      ;RETURN TO 3$ ON INTR
   996  005612  012700  000010                  MOV     #10,R0          ;DO EIGHT 200 WORD XFER'S
   997  005616  012737  000000  001126          MOV     #0,$BDDAT       ;USE $BDDAT AS A COUNTER
   998  005624  012777  177470  173666  1$:     MOV     #-200.,@DRVWCR  ;DO 200 XFERS (DATI'S)
```

```
  999  005632  012777  015400  173662          MOV     #DBUF,@DRVBAR   ;FROM DBUF
 1000  005640  106427  000000                  MTPS    #0              ;ALLOW AN INTR
 1001  005644  012777  000111  173652          MOV     #111,@DRVCSR    ;SET IE, FNCT3 & GO
 1002  005652  052777  000400  173644          BIS     #400,@DRVCSR    ;SET CYCLE
 1003  005660  000240                          NOP                     ;FREEBEE
 1004  005662  005237  001126          2$:     INC     $BDDAT          ;START COUNTING
 1005  005666  001375                          BNE     2$              ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
 1006  005670  012737  004710  001124          MOV     #4710,$GDDAT    ;LD EXPECTED
 1007  005676  017737  173622  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
 1008  005704  042777  000100  173612          BIC     #100,@DRVCSR    ;CLR IE
 1009  005712  104005                          ERROR   5               ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
 1010  005714  000423                          BR      5$              ;GO RESTORE VECTOR
 1011  005716  022626          3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
 1012  005720  005300                          DEC     R0              ;DONE 8 TIMES?
 1013  005722  001340                          BNE     1$              ;BR IF NOT
 1014  005724  022737  000000  001126          CMP     #0,$BDDAT       ;$BDDAT SHOULD HAVE BEEN COUNTED
 1015  005732  103401                          BCS     4$              ;BR IF SO
 1016  005734  104015                          ERROR   15              ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
 1017  005736  017737  173562  001126  4$:     MOV     @DRVCSR,$BDDAT  ;READ STATUS
 1018  005744  012737  004710  001124          MOV     #4710,$GDDAT    ;LD EXPECTED
 1019  005752  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
 1020  005760  001401                          BEQ     5$              ;BR IF SO
 1021  005762  104007                          ERROR   7               ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
 1022  005764  004737  010132          5$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
 1023
 1024                                  ;;*****************************************************************
  (3)                                  ;*TEST 25       TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
  (3)                                  ;;*****************************************************************
  (2)  005770  000004          TST25:  SCOPE
  (1)  005772  012737  000200  001160          MOV     #200,$TIMES     ;;DO 200 ITERATIONS
 1025  006000  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
 1026  006004  006122                          2$                      ;RETURN TO 2$ ON INTR
 1027  006006  004737  010410                  JSR     PC,LDBUF1       ;GO SET UP DBUF (SPECIAL COM PATTERN)
 1028  006012  012737  011702  006130          MOV     #11702,3$       ;3$ CONTAINS EXPECTED STATUS
 1029  006020  012737  000001  006132          MOV     #1,4$           ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
 1030  006026  106427  000000          1$:     MTPS    #0              ;ALLOW INTR
 1031  006032  012777  015400  173462          MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
 1032  006040  013777  006132  173452          MOV     4$,@DRVWCR      ;GET XFER #
 1033  006046  005477  173446                  NEG     @DRVWCR         ;NEGATE FOR WC
 1034  006052  012777  010101  173444          MOV     #10101,@DRVCSR  ;SET UP MAINT, IE & GO
 1035  006060  052777  000400  173436          BIS     #400,@DRVCSR    ;SET CYCLE
 1036  006066  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
 1037  006074  013737  006130  001124          MOV     3$,$GDDAT       ;LD EXPECTED
 1038  006102  017737  173416  001126          MOV     @DRVCSR,$BDDAT  ;READ STATUS
 1039  006110  042777  000100  173406          BIC     #100,@DRVCSR    ;DISABLE IE
 1040  006116  104005                          ERROR   5               ;NO INTR ON XFER (IN MAINT MD)
 1041  006120  000440                          BR      6$              ;GO RESTORE VECTOR
 1042  006122  022626          2$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
 1043  006124  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
 1044  006130  011702          3$:     11702                   ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
 1045  006132  000001          4$:     1                       ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
 1046  006134  104007                  ERROR   7               ;RETURN HERE IF STATUS ER - WILL EXPECT
 1047                                                          ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
 1048                                                          ;CYCLE, READY & IE
 1049  006136  000431                  BR      6$              ;GO RESTORE VECTOR
 1050  006140  104010                  ERROR   10              ;RETURN HERE IF WC ER - SHOULD BE 0
```

```
                                BR      6$              ;GO RESTORE VECTOR
                                ERROR   11              ;RETURN HERE IF BAR ER-
                                BR      6$              ;GO RESTORE VECTOR
006130                          ADD     #1002,3$        ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
006130                          BIT     #BIT13,3$       ;LOOK FOR OVERFLOW
                                BEQ     5$              ;BR IF NOT
006130                          MOV     #10700,3$       ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
006132                  5$:     INC     4$              ;ADVANCE CURRENT XFER #
006132                          CMP     #11,4$          ;HAVE 10 XFERS BEEN DONE
                                BNE     1$              ;BR IF NOT
010526                          JSR     R5,CKDAT1       ;NOW GO CHECK DATA
                                10                      ;# OF XFER'S TO CHECK
                                ERROR   13              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
                                NOP                     ;RESTORE VECTOR NEXT
010132                  6$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR

                        ;;*******************************************************************
                        ;*TEST 26       TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
                        ;;*******************************************************************
                        TST26:  SCOPE
001160                          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
                                MOV     #2,R0           ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
010112                  1$:     JSR     R5,SETVEC       ;GO SET UP INTERRUPT RETURN
                                2$                      ;RETURN TO 2$ ON TIMEOUT INTR
173242                          MOV     #-1,@DRVWCR     ;SET UP FOR ONE XFER'S
173236                          MOV     #160000,@DRVBAR ;SET UP CA TO 160000 (RESERVED)
                                MTPS    #0              ;ALLOW INTR
173226                          MOV     #161,@DRVCSR    ;SET IE, XAD 17,16 & GO
173220                          BIS     #400,@DRVCSR    ;SET CYCLE
001122                          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS - SHOULD NEVER GET HERE
001124                          MOV     #140760,$GDDAT  ;LD EXPECTED
001126                          MOV     @DRVCSR,$BDDAT  ;GIVE THEM THE STATUS
173170                          BIC     #100,@DRVCSR    ;CLR IE
                                ERROR   16              ;NEX FAILED TO CAUSE AN INTERRUPT
                                BR      7$              ;GO RESTORE VECTOR
                        2$:     CMP     (SP)+,(SP)+     ;SHOULD INTR RETURN HERE - FIX STACK
001126                          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
001124                          MOV     #140760,$GDDAT  ;LD EXPECTED
001126                          CMP     $GDDAT,$BDDAT   ;CORRECT?
                                BEQ     3$              ;BR IF SO
001122                          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                ERROR   16              ;STATUS ER - EXPECTED
                                                        ;ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
                                BR      7$              ;GO RESTORE VECTOR
001126                  3$:     MOV     @DRVWCR,$BDDAT  ;READ WORD COUNT
001124                          MOV     #-1,$GDDAT      ;SHOULD STILL HAVE -1
001126                          CMP     $GDDAT,$BDDAT   ;CORRECT?
                                BEQ     4$              ;BR IF SO
001122                          MOV     DRVWCR,$BDADR   ;SET UP WCR ADRS
                                ERROR   10              ;WC INCREMENTED ON A TIMEOUT ER
                                BR      7$              ;GO RESTORE VECTOR
001126                  4$:     MOV     @DRVBAR,$BDDAT  ;READ BUFFER ADRS
001124                          MOV     #160000,$GDDAT  ;SHOULD NOT HAVE INCREMENTED
001126                          CMP     $GDDAT,$BDDAT   ;CORRECT?
                                BEQ     5$              ;BR IF SO
001122                          MOV     DRVBAR,$BDADR   ;SET UP BAR ADRS
```

```
                                        ERROR   11              ;BAR INCREMENTED ON A TIMEOUT ER
                                        BR      7$              ;GO RESTORE VECTOR
                                5$:     DEC     R0              ;KEEP TRACK ON HOW TO CLR
                                        BEQ     6$              ;BR IF CLR BY RESET
                                        BIC     #40000,@DRVCSR  ;WRITE NEX TO ZERO
                                        MOV     @DRVCSR,$BDDAT  ;READ CSR
                                        MOV     #760,$GDDAT     ;LD EXPECTED
                                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                                        BEQ     1$              ;BR IF SO + REPEAT TEST FOR RESET TEST
                                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                        ERROR   16              ;'NEX' FAILED TO WRITE TO ZERO
                                        BR      7$              ;GO RESTORE VECTOR
                                6$:     RESET                   ;ISSUE BUS RESET
                                        MOV     @DRVCSR,$BDDAT  ;READ THE CSR
                                        MOV     #200,$GDDAT     ;EXPECT ONLY READY
                                        CMP     $GDDAT,$BDDAT   ;CORRECT?
                                        BEQ     7$              ;BR IF SO
                                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                        ERROR   16              ;RESET FAILED TO CLR 'NEX'
                                7$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
                                ;;*******************************************************************
                                ;*TEST 27        TEST 200 NPR TRANSFERS IN MAINT MODE
                                ;;*******************************************************************
                                TST27:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        JSR     R5,SETVEC       ;GO SET UP INTR RETURN
                                        2$                      ;RETURN TO 2$ ON INTR
                                        JSR     PC,LDBUF1       ;GO SET UP DBUF (SPECIAL COM PATTERN)
                                        MOV     #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
                                        MOV     #-200.,@DRVWCR  ;SET UP FOR 200 XFER'S
                                        MTPS    #0              ;ALLOW INTR
                                        MOV     #10101,@DRVCSR  ;SET MAINT, IE & GO
                                        BIS     #400,@DRVCSR    ;SET CYCLE
                                        MOV     #0,$BDDAT       ;SET UP A COUNTER
                                1$:     INC     $BDDAT          ;COUNT AWAY
                                        BNE     1$              ;WAIT TILL DONE - SHOULD INTR BEFORE OVFLO
                                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                                        MOV     #10700,$GDDAT   ;LD EXPECTED
                                        MOV     @DRVCSR,$BDDAT  ;READ STATUS
                                        BIC     #100,@DRVCSR    ;DISABLE IE
                                        ERROR   5               ;NO INTR AFTER 200 MAINT MODE XFER'S
                                        BR      3$              ;GO RESTORE VECTOR
                                2$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
                                        JSR     R5,CKSTAT       ;GO CHECK STATUS
                                        10700                   ;EXPECTED STATUS
                                        200.                    ;# OF XFERS
                                        ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
                                                                ;CYCLE, READY & IE
                                        BR      3$              ;GO RESTORE VECTOR
                                        ERROR   10              ;RETURN HERE IF WC ER - SHOULD BE 0
                                        BR      3$              ;GO RESTORE VECTOR
                                        ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
                                        BR      3$              ;GO RESTORE VECTOR
                                        JSR     R5,CKDAT1       ;RETURN HERE IF OK - NOW GO CHECK DATA
                                        200.                    ;# OF XFER'S TO CHECK
                                        ERROR   13              ;RETURN HERE IF DATA ER - (WITH MAINT SET)
```

```
1155  006772  000240                          NOP                     ;RESTORE VECTOR NEXT
1156  006774  004737  010132          3$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR
1157                                  ;;*******************************************************************
 (3)                                  ;*TEST 30       TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
 (3)                                  ;;*******************************************************************
 (2)  007000  000004                  TST30:  SCOPE
 (1)  007002  012737  000005  001160          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1158  007010  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
1159  007014  007150                          3$                      ;RETURN TO 3$ ON INTR
1160  007016  005037  001542                  CLR     DBUFP           ;GET LOWEST BUFFER ADRS
1161  007022  062737  020000  001542  1$:     ADD     #20000,DBUFP    ;POINT TO NEXT 4K
1162  007030  023737  001540  001542          CMP     CORSZ,DBUFP     ;IS THE 4K THERE?
1163  007036  001465                          BEQ     4$              ;BR IF NOT
1164  007040  004737  010410                  JSR     PC,LDBUF1       ;GO SET UP SPECIAL COMPEMENT PATTERN
1165  007044  013777  001542  172450          MOV     DBUFP,@DRVBAR   ;SET UP BUFFER ADRS
1166  007052  012777  177470  172440          MOV     #-200.,@DRVWCR  ;SET UP FOR 200 XFER'S
1167  007060  106427  000000                  MTPS    #0              ;ALLOW INTR
1168  007064  012777  010101  172432          MOV     #10101,@DRVCSR  ;SET MAINT,IE & GO
1169  007072  052777  000400  172424          BIS     #400,@DRVCSR    ;SET CYCLE
1170  007100  012737  000000  001126          MOV     #0,$BDDAT       ;SET UP A COUNTER
1171  007106  005237  001126          2$:     INC     $BDDAT          ;COUNT AWAY
1172  007112  001375                          BNE     2$              ;SHOULD ALWAYS INTR FROM THIS LOOP
1173  007114  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRA
1174  007122  012737  010700  001124          MOV     #10700,$GDDAT   ;LD EXPECTED
1175  007130  017737  172370  001126          MOV     @DRVCSR,$BDDAT  ;READ CSR
1176  007136  042777  000100  172360          BIC     #100,@DRVCSR    ;DISABLE IE
1177  007144  104005                          ERROR   5               ;NO INTR AFTER 200 MAINT MODE XFER'S
1178  007146  000421                          BR      4$              ;GO RESTORE VECTOR
1179  007150  022626                  3$:     CMP     (SP)+,(SP)+     ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1180  007152  004537  010156                  JSR     R5,CKSTAT       ;GO CHECK STATUS
1181  007156  010700                          10700                   ;EXPECTED STATUS
1182  007160  000310                          200.                    ;# OF XFER'S
1183  007162  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
1184                                                                  ;CYCLE, READY & IE
1185  007164  000412                          BR      4$              ;GO RESTORE VECTOR
1186  007166  104010                          ERROR   10              ;RETURN HERE IF WC ER - SHOULD = 0
1187  007170  000410                          BR      4$              ;GO RESTORE VECTOR
1188  007172  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = DBUFP+620
1189  007174  000406                          BR      4$              ;GO RESTORE VECTOR
1190  007176  004537  010526                  JSR     R5,CKDAT1       ;RETURN HERE IF OK - NOW GO CK DATA
1191  007202  000310                          200.                    ;# OF XFER'S TO CK
1192  007204  104013                          ERROR   13              ;RETURN HERE IF DATA ER
1193  007206  000401                          BR      4$              ;GO RESTORE VECTOR
1194  007210  000704                          BR      1$              ;TRY NEXT BANK
1195  007212  012737  015400  001542  4$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
1196  007220  004737  010132                  JSR     PC,RSTVEC       ;GO RESTORE VECTOR
1197                                  ;;*******************************************************************
 (3)                                  ;*TEST 31       TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
 (3)                                  ;;*******************************************************************
 (2)  007224  000004                  TST31:  SCOPE
1198  007226  105777  171706                  TSTB    @SWR            ;KDF11-B?
1199  007232  100507                          BMI     ALTERN          ;IF YES,GO TO THE NEXT SUBTEST
1200  007234  004537  010112          101$:   JSR     R5,SETVEC       ;GO SET UP INTR RETURN
1201  007240  007344                          1$                      ;RETURN TO 1$ ON INTR
1202  007242  012777  177777  172250          MOV     #-1,@DRVWCR     ;SET UP WC - 1 XFER
1203  007250  013737  001150  001542          MOV     $TPS,DBUFP      ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
```

```
1204  007256  013777  001150  172236          MOV     $TPS,@DRVBAR    ;SET UP BUFFER ADRS - FROM PRINTER CSR
1205  007264  106427  000000                  MTPS    #0              ;ALLOW INTR
1206  007270  005077  172232                  CLR     @DRVDBR         ;ZERO THE DBR
1207  007274  012777  000161  172222          MOV     #161,@DRVCSR    ;SET IE, XAD17 & XAD16, & GO
1208  007302  052777  000400  172214          BIS     #400,@DRVCSR    ;SET CYCLE
1209  007310  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
1210  007316  012737  000760  001124          MOV     #760,$GDDAT     ;LD EXPECTED STATUS
1211  007324  017737  172174  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
1212  007332  042777  000100  172164          BIC     #100,@DRVCSR    ;DISABLE IE
1213  007340  104005                          ERROR   5               ;NO INTR ON 1 WD XFER FROM XCSR
1214  007342  000434                          BR      2$              ;GO RESTORE VECTOR
1215  007344  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STK SINCE NO RTI
1216  007346  004537  010156                  JSR     R5,CKSTAT       ;GO CK STATUS
1217  007352  000760                          760                     ;CSR EXPECTED STATUS
1218  007354  000001                          1                       ;# OF XFER'S
1219  007356  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1220                                                                  ;XAD17 & XAD16, RDY & IE
1221  007360  000425                          BR      2$              ;GO RESTORE VECTOR
1222  007362  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
1223  007364  000423                          BR      2$              ;GO RESTORE VECTOR
1224  007366  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
1225  007370  000421                          BR      2$              ;GO RESTORE VECTOR
1226  007372  017737  171552  001124          MOV     @$TPS,$GDDAT    ;RETURN HERE IF OK - GET XCSR CONTENTS
1227  007400  017737  172122  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
1228  007406  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
1229  007414  001407                          BEQ     2$              ;BR IF SO
1230  007416  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
1231  007424  013737  001150  001120          MOV     $TPS,$GDADR     ;GET ADRS OF DATA (XCSR)
1232  007432  104020                          ERROR   20              ;DATA ER FROM TTY PRINTER CSR
1233  007434  012737  015400  001542  2$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
1234  007442  004737  010132                  JSR     PC,RSTVEC       ;GO RESTORE VECTOR
1235  007446  000137  007672                  JMP     NXDEV           ;IGNORE NEXT SUBTEST
1236
1237                                  ;;*****************************************************************
1238                                  ;IN CASE OF KDF11-B TEST SOME SPECIFIED I/O PAGE
1239                                  ;;*****************************************************************
1240  007452  005737  001544          ALTERN: TST     IOPAGE          ;I/O TRANSFER DESIRED?
1241  007456  001505                          BEQ     NXDEV           ;IF NOT, SKIP THE TEST
1242  007460  004537  010112                  JSR     R5,SETVEC       ;GO SET UP INTR RETURN
1243  007464  007570                          1$                      ;RETURN TO 1$ ON INTR
1244  007466  012777  177777  172024          MOV     #-1,@DRVWCR     ;SET UP WC = 1 XFER
1245  007474  013737  001544  001542          MOV     IOPAGE,DBUFP    ;SET UP BUFFER ADRS TO I/O CSR ADRS
1246  007502  013777  001544  172012          MOV     IOPAGE,@DRVBAR  ;SET UP BUFFER ADRS - FROM I/O CSR
1247  007510  106427  000000                  MTPS    #0              ;ALLOW INTR
1248  007514  005077  172006                  CLR     @DRVDBR         ;ZERO THE DBR
1249  007520  012777  000161  171776          MOV     #161,@DRVCSR    ;SET IE, XAD17 & XAD16, & GO
1250  007526  052777  000400  171770          BIS     #400,@DRVCSR    ;SET CYCLE
1251  007534  013737  001524  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
1252  007542  012737  000760  001124          MOV     #760,$GDDAT     ;LD EXPECTED STATUS
1253  007550  017737  171750  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
1254  007556  042777  000100  171740          BIC     #100,@DRVCSR    ;DISABLE IE
1255  007564  104005                          ERROR   5               ;NO INTR ON 1 WD XFER FROM XCSR
1256  007566  000434                          BR      2$              ;GO RESTORE VECTOR
1257  007570  022626                  1$:     CMP     (SP)+,(SP)+     ;INTR RETURNS HERE - FIX STK SINCE NO RTI
1258  007572  004537  010156                  JSR     R5,CKSTAT       ;GO CK STATUS
1259  007576  000760                          760                     ;CSR EXPECTED STATUS
```

```
1260  007600  000001                          1                       ;# OF XFER'S
1261  007602  104007                          ERROR   7               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1262                                                                  ;XAD17 & XAD16, RDY & IE
1263  007604  000425                          BR      2$              ;GO RESTORE VECTOR
1264  007606  104010                          ERROR   10              ;RETURN HERE IF WC ER - EXPECTED 0
1265  007610  000423                          BR      2$              ;GO RESTORE VECTOR
1266  007612  104011                          ERROR   11              ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
1267  007614  000421                          BR      2$              ;GO RESTORE VECTOR
1268  007616  017737  171722  001124          MOV     @IOPAGE,$GDDAT  ;RETURN HERE IF OK - GET XCSR CONTENTS
1269  007624  017737  171676  001126          MOV     @DRVDBR,$BDDAT  ;READ DBR
1270  007632  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
1271  007640  001407                          BEQ     2$              ;BR IF SO
1272  007642  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
1273  007650  013737  001544  001120          MOV     IOPAGE,$GDADR   ;GET ADRS OF DATA (XCSR)
1274  007656  104020                          ERROR   20              ;DATA ER FROM TTY PRINTER CSR
1275  007660  012737  015400  001542  2$:     MOV     #DBUF,DBUFP     ;RESTORE BUFFER ADRS TO LOWEST 4K
1276  007666  004737  010132                  JSR     PC,RSTVEC       ;GO RESTORE VECTOR
1277
1278                                          ;;**************************************************************
1279                                          ;DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
1280                                          ;;**************************************************************
1281  007672  000004                  NXDEV:  SCOPE
1282  007674  000241                  NXDEV1: CLC                     ;CLR CARRY
1283  007676  006037  001536                  ROR     DMAP            ;LOOK FOR NEXT
1284  007702  001432                          BEQ     $EOP            ;BR IF ALL TESTED
1285  007704  062737  000010  001520          ADD     #10,DRVWCR      ;OFFSET BASE BUS ADRS TO NEXT DRV11B
1286  007712  062737  000010  001522          ADD     #10,DRVBAR      ;
1287  007720  062737  000010  001524          ADD     #10,DRVCSR      ;
1288  007726  062737  000010  001526          ADD     #10,DRVDBR      ;
1289  007734  062737  000004  001530          ADD     #4,DRVCT0       ;OFFSET VECTOR ADRS TO NEXT
1290  007742  062737  000004  001532          ADD     #4,DRVCT2       ;
1291  007750  005237  001206                  INC     $UNIT           ;COUNT DEVICE
1292  007754  032737  000001  001536          BIT     #1,DMAP         ;IS IT SELECTED?
1293  007762  001744                          BEQ     NXDEV1          ;BR IF NOT
1294  007764  000137  002406                  JMP     RESTRT          ;TEST NEXT
1295                                  .SBTTL  END OF PASS ROUTINE
 (1)
 (2)                                      ;;**************************************************************
 (1)                                      ;*INCREMENT THE PASS NUMBER ($PASS)
 (1)                                      ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 (1)                                      ;*IF THERES A MONITOR GO TO IT
 (1)                                      ;*IF THERE ISN'T JUMP TO START1
 (1)
 (1)  007770                      $EOP:
 (2)  007770  000240                      NOP
 (1)  007772  005037  001102              CLR     $TSTNM          ;;ZERO THE TEST NUMBER
 (1)  007776  005037  001160              CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
 (1)  010002  005237  001202              INC     $PASS           ;;INCREMENT THE PASS NUMBER
 (1)  010006  042737  100000  001202      BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
 (1)  010014  005327                      DEC     (PC)+           ;;LOOP?
 (1)  010016  000001              $EOPCT: .WORD   1
 (1)  010020  003022                      BGT     $DOAGN          ;;YES
 (1)  010022  012737                      MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
 (1)  010024  000001              $ENDCT: .WORD   1
 (1)  010026  010016                      $EOPCT
 (1)  010030  104401  010075              TYPE    ,$ENDMG         ;;TYPE "END PASS #"
```

```
(2)  010034  013746  001202            MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
(2)  010040  104405                    TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1)  010042  104401  010072            TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
(1)  010046  013700  000042  $GET42:   MOV     @#42,R0         ;;GET MONITOR ADDRESS
(1)  010052  001405                    BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
(1)  010054  000005                    RESET                   ;;CLEAR THE WORLD
(1)  010056  004710            $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
(1)  010060  000240                    NOP                     ;;SAVE ROOM
(1)  010062  000240                    NOP                     ;;FOR
(1)  010064  000240                    NOP                     ;;ACT11
(1)  010066                    $DOAGN:
(1)  010066  000137                    JMP     @(PC)+          ;;RETURN
(1)  010070  002012            $RTNAD: .WORD   START1
(1)  010072     377     377     000    $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
(1)  010075     015  042412  042116    $ENDMG: .ASCIZ  <15><12>/END PASS #/
(1)  010102  050040  051501  020123
(1)  010 10  000043
```

```
                                        .SBTTL  PROGRAM SUBROUTINES


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

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

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

```
                                ;;*****************************************************************
                                ;THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
                                ;FOR 199 LOCATIONS - THE LAST LOCATION IS LOADED WITH THE
                                ;#70707 WHICH SHOULD BE THE DATA WORD AVAILABLE IN THE DBR
                                ;AT THE COMPLETION OF A 200 WORD TRANSFER
                                ;;*****************************************************************
010336  012703  015400        LDBUF:  MOV     #DBUF,R3        ;GET BUFFER ADRS
010342  012704  177776        1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
010346  010423                2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
010350  005104                        COM     R4              ;MAKE INTO FLOATING 1
010352  022703  016216                CMP     #DBUF+616,R3    ;AT END OF BUFFER?
010356  001003                        BNE     4$              ;BR IF NOT
010360  012713  070707        3$:     MOV     #70707,(R3)     ;LOAD LAST DATVAR (SPECIAL)
010364  000207                        RTS     PC              ;GET OUT
010366  010423                4$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 1)
010370  022703  016216                CMP     #DBUF+616,R3    ;AT END OF BUFFER?
010374  001771                        BEQ     3$              ;BR IF SO
010376  005104                        COM     R4              ;BACK TO FLOATING ZERO
010400  006304                        ASL     R4              ;SHIFT LEFT
010402  005204                        INC     R4              ;KEEP LSB SET
010404  103356                        BCC     1$              ;GO RESET FLOATING PATRN
010406  000757                        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
                                ;;*****************************************************************
010410  013703  001542        LDBUF1: MOV     DBUFP,R3        ;GET BUFFER ADRS
010414  010305                        MOV     R3,R5           ;SAVE IN R5
010416  062705  000620                ADD     #620,R5         ;POINT TO END OF BUFFER
010422  012704  177776        1$:     MOV     #177776,R4      ;SET UP FLOATING ZERO PATRN
010426  010423                2$:     MOV     R4,(R3)+        ;LOAD IT (FLOATING 0)
010430  005023                        CLR     (R3)+           ;ZERO NEXT
010432  005104                        COM     R4              ;SET UP FLOATING 1
010434  010423                        MOV     R4,(R3)+        ;LOAD IT
010436  005023                        CLR     (R3)+           ;ZERO NEXT
010440  020503                        CMP     R5,R3           ;200 LOCS DONE?
010442  001001                        BNE     3$              ;BR IF NOT
010444  000207                        RTS     PC              ;GET OUT
010446  005104                3$:     COM     R4              ;BACK TO FLOATING ZERO
010450  006304                        ASL     R4              ;SHIFT LEFT
010452  005204                        INC     R4              ;KEEP LSB SET
010454  103362                        BCC     1$              ;GO RESET FLOATING PATRN
010456  000763                        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
                                ;;*****************************************************************
010460  012701  015400        CKDAT:  MOV     #DBUF,R1        ;GET BUFFER ADRS
010464  022721  177377        1$:     CMP     #177377,(R1)+   ;DATA OK?
010470  001410                        BEQ     2$              ;BR IF SO
010472  013737  001526  001122        MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
```

```
1409  010500  014137  001126                  MOV     -(R1),$BDDAT    ;GET ACTUAL DATA XFERED
1410  010504  010137  001120                  MOV     R1,$GDADR       ;GET MEMORY ADRS
1411  010510  000207                          RTS     PC              ;RETURN TO ERROR
1412  010512  022701  016220          2$:     CMP     #DBUF+620,R1    ;AT END OF 'DBUF'?
1413  010516  001362                          BNE     1$              ;BR IF MORE
1414  010520  062716  000002                  ADD     #2,(SP)         ;ADJUST STACK FOR GOOD RETURN
1415  010524  000207                          RTS     PC              ;GET OUT
1416
1417                                  ;*****************************************************************
1418                                  ;THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
1419                                  ;DATA (177776,177776,1,1,177775,177775,2,2,177773,177773,ETC.)
1420                                  ;ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED
1421                                  ;BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 -
1422                                  ;IF NO ERROR THE RETURN IS TO CALL +10
1423                                  ;;****************************************************************
1424  010526  012500                  CKDAT1: MOV     (R5)+,R0        ;GET # OF CHECKS
1425  010530  013702  001542                  MOV     DBUFP,R2        ;GET BUFFER ADRS
1426  010534  012701  177776          1$:     MOV     #177776,R1      ;SET UP FLOATING ZERO PATRN
1427  010540  005003                          CLR     R3              ;R3 SAYS WHEN TO SHIFT PATRN
1428  010542  020122                  2$:     CMP     R1,(R2)+        ;DATA OK?
1429  010544  001010                          BNE     3$              ;BR IF NOT
1430  010546  020122                          CMP     R1,(R2)+        ;DATA WRITTEN OK?
1431  010550  001006                          BNE     3$              ;BR IF NOT
1432  010552  162700  000002                  SUB     #2,R0           ;ACCOUNT FOR TWO ADRS'S
1433  010556  003015                          BGT     4$              ;BR IF MORE
1434  010560  062705  000004                  ADD     #4,R5           ;ADJUST FOR GOOD RETURN
1435  010564  000205                          RTS     R5              ;EXIT
1436  010566  014237  001126          3$:     MOV     -(R2),$BDDAT    ;GET BAD DATA
1437  010572  010237  001120                  MOV     R2,$GDADR       ;GET MEM ADRS
1438  010576  010137  001124                  MOV     R1,$GDDAT       ;LD EXPECTED DATA
1439  010602  013737  001526  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
1440  010610  000205                          RTS     R5              ;RETURN TO ERROR
1441  010612  005101                  4$:     COM     R1              ;NOW EXPECT COMPLEMENT
1442  010614  005103                          COM     R3              ;TIME TO SHIFT?
1443  010616  001351                          BNE     2$              ;BR IF NOT
1444  010620  006301                          ASL     R1              ;SHIFT LEFT
1445  010622  005201                          INC     R1              ;KEEP LSB SET
1446  010624  103343                          BCC     1$              ;GO RESET FLOATING PATRN
1447  010626  000745                          BR      2$              ;DO NEXT
```

```
1449                                    .SBTTL  SYSMAC ROUTINES
1450
1451                                    .SBTTL  TYPE ROUTINE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 (1)                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 (1)                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 (1)                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 (1)                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 (1)                                    ;*
 (1)                                    ;*CALL:
 (1)                                    ;*1) USING A TRAP INSTRUCTION
 (1)                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 (1)                                    ;*OR
 (1)                                    ;*      TYPE
 (1)                                    ;*      MESADR
 (1)                                    ;*
 (1)
 (1)  010630  105737  001157            $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
 (1)  010634  100002                            BPL     1$              ;;BR IF YES
 (1)  010636  000000                            HALT                    ;;HALT HERE IF NO TERMINAL
 (1)  010640  000430                            BR      3$              ;;LEAVE
 (1)  010642  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
 (1)  010644  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
 (1)  010650  122737  000001  001214            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 (1)  010656  001011                            BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
 (1)  010660  132737  000100  001215            BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
 (1)  010666  001405                            BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
 (1)  010670  010037  010700                    MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
 (1)  010674  004737  011172                    JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
 (1)  010700  000000                    61$:    .WORD   0               ;;MESSAGE ADDRESS
 (1)  010702  132737  000040  001215    62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
 (1)  010710  001003                            BNE     60$             ;;YES,SKIP TYPE OUT
 (1)  010712  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 (1)  010714  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 (1)  010716  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 (1)  010720  012600                    60$:    MOV     (SP)+,R0        ;;RESTORE R0
 (1)  010722  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
 (1)  010726  000002                            RTI                     ;;RETURN
 (1)  010730  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 (1)  010734  001430                            BEQ     8$
 (1)  010736  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 (1)  010742  001006                            BNE     5$
 (1)  010744  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
 (1)  010746  104401                            TYPE                    ;;TYPE A CR AND LF
 (1)  010750  001171                            $CRLF
 (1)  010752  105037  011160                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
 (1)  010756  000755                            BR      2$              ;;GET NEXT CHARACTER
 (1)  010760  004737  011042            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 (1)  010764  123726  001156            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 (1)  010770  001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
 (1)  010772  013746  001154                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 (1)                                            ;;AND THE NULL CHAR.
 (1)  010776  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 (1)  011002  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1)  011004  004737  011042              JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  011010  105337  011160              DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  011014  000770                      BR      7$              ;;LOOP
(1)
(1)                                  ;HORIZONTAL TAB PROCESSOR
(1)
(1)  011016  112716  000040          8$: MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  011022  004737  011042          9$: JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  011026  132737  000007  011160      BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  011034  001372                      BNE     9$              ;;TAB STOP
(1)  011036  005726                      TST     (SP)+           ;;POP SPACE OFF STACK
(1)  011040  000724                      BR      2$              ;;GET NEXT CHARACTER
(1)  011042                          $TYPEC:
(1)  011042  105777  170076              TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
(1)  011046  100022                      BPL     10$             ;;BR IF NOT                     ;MJD001
(1)  011050  017746  170072              MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
(1)  011054  042716  177600              BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
(1)  011060  122716  000023              CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
(1)  011064  001012                      BNE     102$            ;;BR IF NOT                     ;MJD001
(1)  011066                          101$:                                                       ;MJD001
(1)  011066  105777  170052              TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
(1)  011072  100375                      BPL     101$                                            ;MJD001
(1)  011074  117716  170046              MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
(1)  011100  042716  177600              BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
(1)  011104  122716  000021              CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
(1)  011110  001366                      BNE     101$            ;;BR IF NOT                     ;MJD001
(1)  011112                          102$:                                                       ;MJD001
(1)  011112  005726                      TST     (SP)+           ;;FIX STACK                     ;MJD001
(1)  011114                          10$:                                                        ;MJD001
(1)  011114  105777  170030              TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  011120  100375                      BPL     10$                                             ;MJD001
(1)  011122  116677  000002  170022      MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  011130  122766  000015  000002      CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  011136  001003                      BNE     1$              ;;BRANCH IF NO
(1)  011140  105037  011160              CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  011144  000406                      BR      $TYPEX          ;;EXIT
(1)  011146  122766  000012  000002  1$: CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  011154  001402                      BEQ     $TYPEX          ;;BRANCH IF YES
(1)  011156  105227                      INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  011160  000000                  $CHARCNT:.WORD  0           ;;CHARACTER COUNT STORAGE
(1)  011162  000207                  $TYPEX: RTS     PC
(1)
1452                                 .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)  011164  112737  000001  011430  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  011172  112737  000001  011426  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  011200  000403                      BR      $ATYC
(1)  011202  112737  000001  011430  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  011210                          $ATYC:
(3)  011210  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  011212  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  011214  105737  011426              TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  011220  001450                      BEQ     5$              ;;IF NOT:  BR
(1)  011222  122737  000001  001214      CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  011230  001031                      BNE     3$              ;;IF NOT:  BR
```

```
(1)  011232  132737  000100  001215          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  011240  001425                          BEQ     3$              ;;IF NOT:  BR
(1)  011242  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  011246  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  011254  005737  001174          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  011260  001375                          BNE     1$              ;;IF NOT:  WAIT
(1)  011262  010037  001210                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  011266  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  011270  001376                          BNE     2$
(1)  011272  163700  001210                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  011276  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  011300  010037  001212                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  011304  012737  000004  001174          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  011312  000413                          BR      5$
(1)  011314  017637  000004  011340  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN ISR LINKAGE
(1)  011322  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  011330  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  011334  004737  010630                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  011340  000000                  4$:     .WORD   0
(1)  011342                          5$:
(1)  011342  105737  011430          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  011346  001416                          BEQ     12$             ;;IF NOT:  BR
(1)  011350  005737  001214                  TST     $ENV            ;;RUNNING UNDER APT?
(1)  011354  001413                          BEQ     12$             ;;IF NOT:  BR
(1)  011356  005737  001174          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  011362  001375                          BNE     11$             ;;IF NOT:  WAIT
(1)  011364  017637  000004  001176          MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  011372  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  011400  005237  001174                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  011404  105037  011430          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  011410  105037  011427                  CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  011414  105037  011426                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  011420  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  011422  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  011424  000207                          RTS     PC              ;;RETURN
(1)  011426     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  011427     000          $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  011430     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          011432                  .EVEN
(1)          000200          APTSIZE=200
(1)          000001          APTENV=001
(1)          000100          APTSPOOL=100
(1)          000040          APTCSUP=040
1453                         .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                          ;;**********************************************************************
(1)                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                          ;*CALL:
(1)                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                          ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                          ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                          ;*      .BYTE   M               ;;M=1 OR 0
(1)                          ;*                                      ;;1=TYPE LEADING ZEROS
(1)                          ;*                                      ;;0=SUPPRESS LEADING ZEROS
```

```
(1)                                     ;*
(1)                                     ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                     ;*$TYPOS OR $TYPOC
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                     ;*
(1)                                     ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  011432  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  011436  116637  000001  011655          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  011444  112637  011657                   MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  011450  062716  000002                   ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  011454  000406                           BR      $TYPON
(1)  011456  112737  000001  011655  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  011464  112737  000006  011657          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  011472  112737  000005  011654  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  011500  010346                           MOV     R3,-(SP)        ;;SAVE R3
(1)  011502  010446                           MOV     R4,-(SP)        ;;SAVE R4
(1)  011504  010546                           MOV     R5,-(SP)        ;;SAVE R5
(1)  011506  113704  011657                   MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  011512  005404                           NEG     R4
(1)  011514  062704  000006                   ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  011520  110437  011656                   MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  011524  113704  011655                   MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  011530  016605  000012                   MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  011534  005003                           CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  011536  006105                   1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  011540  000404                           BR      3$              ;;GO DO MSB
(1)  011542  006105                   2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  011544  006105                           ROL     R5
(1)  011546  006105                           ROL     R5
(1)  011550  010503                           MOV     R5,R3
(1)  011552  006103                   3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  011554  105337  011656                   DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  011560  100016                           BPL     7$              ;;BR IF NO
(1)  011562  042703  177770                   BIC     #177770,R3      ;;GET RID OF JUNK
(1)  011566  001002                           BNE     4$              ;;TEST FOR 0
(1)  011570  005704                           TST     R4              ;;SUPPRESS THIS 0?
(1)  011572  001403                           BEQ     5$              ;;BR IF YES
(1)  011574  005204                   4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  011576  052703  000060                   BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  011602  052703  000040           5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  011606  110337  011652                   MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  011612  104401  011652                   TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  011616  105337  011654           7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  011622  003347                           BGT     2$              ;;BR IF MORE TO DO
(1)  011624  002402                           BLT     6$              ;;BR IF DONE
(1)  011626  005204                           INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  011630  000744                           BR      2$              ;;GO DO THE LAST DIGIT
(1)  011632  012605                   6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  011634  012604                           MOV     (SP)+,R4        ;;RESTORE R4
(1)  011636  012603                           MOV     (SP)+,R3        ;;RESTORE R3
```

```
    (1)  011640  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
    (1)  011646  012616                           MOV     (SP)+,(SP)
    (1)  011650  000002                           RTI                     ;;RETURN
    (1)  011652     000           8$:             .BYTE   0               ;;STORAGE FOR ASCII DIGIT
    (1)  011653     000                           .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
    (1)  011654     000           $OCNT:          .BYTE   0               ;;OCTAL DIGIT COUNTER
    (1)  011655     000           $OFILL:         .BYTE   0               ;;ZERO FILL SWITCH
    (1)  011656  000000           $OMODE:         .WORD   0               ;;NUMBER OF DIGITS TO TYPE
  1454                            .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
    (1)
    (2)                           ;;*****************************************************************
    (1)                           ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
    (1)                           ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
    (1)                           ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
    (1)                           ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
    (1)                           ;*REPLACED WITH SPACES.
    (1)                           ;*CALL:
    (1)                           ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
    (1)                           ;*      TYPDS                   ;;GO TO THE ROUTINE
    (1)
    (1)  011660                   $TYPDS:
    (3)  011660  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
    (3)  011662  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    (3)  011664  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
    (3)  011666  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
    (3)  011670  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
    (1)  011672  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
    (1)  011676  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
    (1)  011702  100004                           BPL     1$              ;;BR IF INPUT IS POS.
    (1)  011704  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
    (1)  011706  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
    (1)  011714  005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
    (1)  011716  012703  012074                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
    (1)  011722  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
    (1)  011726  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
    (1)  011730  016001  012064                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
    (1)  011734  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
    (1)  011736  002402                           BLT     4$              ;;BR IF DONE
    (1)  011740  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
    (1)  011742  000774                           BR      3$
    (1)  011744  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
    (1)  011746  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
    (1)  011750  001002                           BNE     5$              ;;FALL THROUGH IF 0
    (1)  011752  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
    (1)  011754  100407                           BMI     7$              ;;BR IF YES
    (1)  011756  106316                   5$:     ASLB    (SP)            ;;MSD?
    (1)  011760  103003                           BCC     6$              ;;BR IF NO
    (1)  011762  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
    (1)  011770  052702  000060           6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
    (1)  011774  052702  000040           7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
    (1)  012000  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
    (1)  012002  005720                           TST     (R0)+           ;;JUST INCREMENTING
    (1)  012004  020027  000010                   CMP     R0,#10          ;;CHECK THE TABLE INDEX
    (1)  012010  002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
    (1)  012012  003002                           BGT     8$              ;;GO TO EXIT
    (1)  012014  010502                           MOV     R5,R2           ;;GET THE LSD
```

```
(1)  012016  000764                          BR      6$              ;;GO CHANGE TO ASCII
(1)  012020  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  012022  100003                          BPL     9$              ;;BR IF NO
(1)  012024  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  012032  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  012034  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  012036  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  012040  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  012042  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  012044  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  012046  104401  012074                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  012052  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  012060  012616                          MOV     (SP)+,(SP)
(1)  012062  000002                          RTI                     ;;RETURN TO USER
(1)  012064  023420                  $DTBL:  10000.
(1)  012066  001750                          1000.
(1)  012070  000144                          100.
(1)  012072  000012                          10.
(1)  012074  000004                  $DBLK:  .BLKW   4
1455                                 .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)                                  ;;**************************************************************************
(1)                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)                                  ;*AND GO TO SWRCK ON ERROR
(1)                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                  ;*SW15=1        HALT ON ERROR
(1)                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
(1)                                  ;*SW10=1        BELL ON ERROR
(1)                                  ;*SW09=1        LOOP ON ERROR
(1)                                  ;*CALL
(1)                                  ;*      ERROR   N       ;;ERROR=EMT AND N-ERROR ITEM NUMBER
(1)
(1)  012104                          $ERROR:
(1)  012104  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(2)  012106  104407                          CKSWR           ;GO LOOK FOR SWR CHANGE
(1)  012110  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  012114  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  012116  013777  001102  167016          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  012124  032777  002000  167006          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
(1)  012132  001402                          BEQ     1$              ;;NO - SKIP
(1)  012134  104401  001164                  TYPE    ,$BELL          ;;RING BELL
(1)  012140  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
(1)  012144  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  012150  162737  000002  001116          SUB     #2,$ERRPC
(1)  012156  117737  166734  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  012164  032777  020000  166746          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  012172  001004                          BNE     20$             ;;SKIP TYPEOUTS
(1)  012174  004737  012274                  JSR     PC,SWRCK        ;;GO TO USER ERROR ROUTINE
(1)  012200  104401  001171                  TYPE    ,$CRLF
(1)  012204                          20$:
(1)  012204  122737  000001  001214          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  012212  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
(1)  012214  113737  001114  012226          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  012222  004737  011202                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
(1)  012226     000                  21$:    .BYTE   0
```

```
(1)  012227  000                        .BYTE   0
(1)  012230  000777                22$:  BR      22$               ;;APT ERROR LOOP
(1)  012232  005777  166702        2$:   TST     @SWR              ;;HALT ON ERROR
(1)  012236  100002                       BPL     3$               ;;SKIP IF CONTINUE
(1)  012240  000000                       HALT                     ;;HALT ON ERROR!
(1)  012242  104407                       CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(1)  012244  032777  001000  166666  3$:   BIT     #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
(1)  012252  001402                       BEQ     4$               ;;BR IF NO
(1)  012254  013716  001110               MOV     $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
(1)  012260  005737  001162        4$:   TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
(1)  012264  001402                       BEQ     5$               ;;BR IF NONE
(1)  012266  013716  001162               MOV     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  012272                        5$:
(1)  012272  000002                       RTI                      ;;RETURN
1456                               ;;*********************************************************************
1457                               ;GO TYPE ERROR
1458                               ;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
1459                               ;;*********************************************************************
1460  012274  113737  001102  001534  SWRCK:  MOVB    $TSTNM,TSTNUM    ;SET UP TEST # ON ER
1461  012302  004737  012312          JSR     PC,$ERRTYP       ;GO TYPE ERROR
1462  012306  104407                   CKSWR                    ;GO LOOK FOR SWR CHANGE
1463  012310  000207                   RTS     PC               ;RETURN TO ERROR HANDLER
1464                              .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                ;;*********************************************************************
(1)                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  012312                        $ERRTYP:
(1)  012312  104401  001171               TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012316  010046                       MOV     R0,-(SP)         ;;SAVE R0
(1)  012320  005000                       CLR     R0               ;;PICKUP THE ITEM INDEX
(1)  012322  153700  001114               BISB    @#$ITEMB,R0
(1)  012326  001004                       BNE     1$               ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                 ;;TYPE THE PC OF THE ERROR
(2)  012330  013746  001116               MOV     $ERRPC,-(SP)     ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                 ;;ERROR ADDRESS
(2)  012334  104402                       TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012336  000426                       BR      6$               ;;GET OUT
(1)  012340  005300                1$:    DEC     R0               ;;ADJUST THE INDEX SO THAT IT WILL
(1)  012342  006300                       ASL     R0               ;;       WORK FOR THE ERROR TABLE
(1)  012344  006300                       ASL     R0
(1)  012346  006300                       ASL     R0
(1)  012350  062700  001320               ADD     #$ERRTB,R0       ;;FORM TABLE POINTER
(1)  012354  012037  012364               MOV     (R0)+,2$         ;;PICKUP "ERROR MESSAGE" POINTER
(1)  012360  001404                       BEQ     3$               ;;SKIP TYPEOUT IF NO POINTER
(1)  012362  104401                       TYPE                     ;;TYPE THE "ERROR MESSAGE"
(1)  012364  000000                2$:    .WORD   0                ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  012366  104401  001171               TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012372  012037  012402        3$:    MOV     (R0)+,4$         ;;PICKUP "DATA HEADER" POINTER
(1)  012376  001404                       BEQ     5$               ;;SKIP TYPEOUT IF 0
(1)  012400  104401                       TYPE                     ;;TYPE THE "DATA HEADER"
(1)  012402  000000                4$:    .WORD   0                ;;"DATA HEADER" POINTER GOES HERE
(1)  012404  104401  001171               TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012410  011000                5$:    MOV     (R0),R0          ;;PICKUP "DATA TABLE" POINTER
```

```
(1)  012412  001004                    BNE     7$              ;;GO TYPE THE DATA
(1)  012414  012600              6$:   MOV     (SP)+,R0        ;;RESTORE R0
(1)  012416  104401  001171            TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & ''LINE FEED''
(1)  012422  000207                    RTS     PC              ;;RETURN
(1)  012424                      7$:
(2)  012424  013046                    MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  012426  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012430  005710                    TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  012432  001770                    BEQ     6$              ;;BR IF NO
(1)  012434  104401  012442            TYPE    ,8$             ;;TYPE TWO(2) SPACES
(1)  012440  000771                    BR      7$              ;;LOOP
(1)  012442  020040      000     8$:   .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          012446                    .EVEN
1465                             .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                              ;;*****************************************************************
(1)                              ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                              ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                              ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                              ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                              ;*SW14=1        LOOP ON TEST
(1)                              ;*SW11=1        INHIBIT ITERATIONS
(1)                              ;*SW09=1        LOOP ON ERROR
(1)                              ;*SW08=1        LOOP ON TEST IN SWR<5:0>
(1)                              ;*CALL
(1)                              ;*      SCOPE           ;;SCOPE=IOT
(1)
(1)  012446                      $SCOPE:
(1)  012446  104407                    CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(2)  012450  104407                    CKSWR                   ;GO LOOK FOR SWR CHANGE
(1)  012452  032777  040000  166460 1$:   BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  012460  001117                    BNE     $OVER           ;;YES IF SW14=1
(1)                              ;#####START OF CODE FOR THE XOR TESTER#####
(1)  012462  000416              $XTSTR: BR     6$               ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
(1)                                                             ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
(1)  012464  013746  000004            MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  012470  012737  012510  000004    MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  012476  005737  177060            TST     @#177060        ;;TIME OUT ON XOR?
(1)  012502  012637  000004            MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  012506  000466                    BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  012510  022626              5$:   CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
(1)  012512  012637  000004            MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  012516  000426                    BR      7$              ;;LOOP ON THE PRESENT TEST
(1)  012520                      6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  012520  032777  000400  166412    BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  012526  001407                    BEQ     2$              ;;BR IF NO
(1)  012530  017746  166404            MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
(1)  012534  042716  000300            BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
(1)  012540  122637  001102            CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
(1)  012544  001465                    BEQ     $OVER           ;;BR IF YES
(1)  012546  105737  001103      2$:   TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
(1)  012552  001421                    BEQ     3$              ;;BR IF NO
(1)  012554  123737  001115  001103    CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  012562  101015                    BHI     3$              ;;BR IF NO
(1)  012564  032777  001000  166346    BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  012572  001404                    BEQ     4$              ;;BR IF NO
```

```
(1)  012574  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  012602  000446                          BR      $OVER
(1)  012604  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
(1)  012610  005037  001160                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  012614  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  012616  032777  004000  166314  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  012624  001011                          BNE     1$              ;;BR IF YES
(1)  012626  005737  001202                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  012632  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  012634  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  012640  023737  001160  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  012646  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  012650  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
(1)  012656  013737  012734  001160          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  012664  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
(1)  012670  113737  001102  001200          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  012676  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  012702  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  012706  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  012712  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  012720  013777  001102  166214  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  012726  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  012732  000002                          RTI                     ;;FIXES PS
(1)  012734  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
1466                                 .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  .ENABL  LSB
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1)                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1)                                  ;*WHEN OPERATING IN TTY FLAG MODE.
(1)  012736  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
(1)  012744  001074                          BNE     15$             ;;BRANCH IF NO
(1)  012746  105777  166172                  TSTB    @$TKS           ;;CHAR THERE?
(1)  012752  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
(1)  012754  117746  166166                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
(1)  012760  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
(1)  012764  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
(1)  012770  001062                          BNE     15$             ;;NO, RETURN TO USER
(1)  012772  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  013000  001456                          BEQ     15$             ;;BRANCH IF YES
(1)
(1)  013002  104401  013473                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  013006  104401  013500          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  013012  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  013016  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  013020  104401  013511                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  013024  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  013026  005046                          CLR     -(SP)           ;;THE NEW SWR
(1)  013030  105777  166110          7$:     TSTB    @$TKS           ;;CHAR THERE?
(1)  013034  100375                          BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  013036  117746  166104                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
```

```
(1)  013042  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  013046  021627  000025          9$:   CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  013052  001005                        BNE     10$             ;;BRANCH IF NOT
(1)  013054  104401  013466                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  013060  062706  000006          20$:  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  013064  000757                        BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  013066  021627  000015          10$:  CMP     (SP),#15        ;;IS IT A <CR>?
(1)  013072  001022                        BNE     16$             ;;BRANCH IF NO
(1)  013074  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  013100  001403                        BEQ     11$             ;;BRANCH IF YES
(1)  013102  016677  000002  166030        MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  013110  062706  000006          11$:  ADD     #6,SP           ;;CLEAR UP STACK
(1)  013114  104401  001171          14$:  TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  013120  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  013126  001003                        BNE     15$             ;;BRANCH IF NOT
(1)  013130  012777  000100  166006        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  013136  000002                  15$:  RTI                     ;;RETURN
(1)  013140  004737  011042          16$:  JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  013144  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
(1)  013150  002420                        BLT     18$             ;;BRANCH IF YES
(1)  013152  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
(1)  013156  003015                        BGT     18$             ;;BRANCH IF YES
(1)  013160  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  013164  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  013170  001403                        BEQ     17$             ;;BRANCH IF YES
(1)  013172  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  013174  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  013176  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  013200  005266  000002          17$:  INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  013204  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  013210  000707                        BR      7$              ;;GET THE NEXT ONE
(1)  013212  104401  001170          18$:  TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  013216  000720                        BR      20$             ;;SIMULATE CONTROL-U
(1)                                 .DSABL  LSB
(1)
(1)
(2)                                 ;;******************************************************************
(1)                                 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                 ;*CALL:
(1)                                 ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                 ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                 ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                 ;
(1)
(1)  013220  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  013222  016666  000004  000002        MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  013230  105777  165710          1$:   TSTB    @$TKS           ;;WAIT FOR
(1)  013234  100375                        BPL     1$              ;;A CHARACTER
(1)  013236  117766  165704  000004        MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  013244  042766  177600  000004        BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  013252  026627  000004  000023        CMP     4(SP),#23       ;;IS IT A CONTROL-S?
```

```
(1)  013260  001013                        BNE     3$              ;;BRANCH IF NO
(1)  013262  105777  165656          2$:    TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  013266  100375                        BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  013270  117746  165652                MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  013274  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  013300  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  013304  001366                        BNE     2$              ;;IF NOT DISCARD IT
(1)  013306  000750                        BR      1$              ;;YES, RESUME
(1)  013310  026627  000004  000021   3$:    CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?                   ;RAN001
(1)  013316  001744                        BEQ     1$              ;;BRANCH IF YES                         ;RAN001
(1)  013320  026627  000004  000140        CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  013326  002407                        BLT     /$              ;;BRANCH IF YES
(1)  013330  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  013336  003003                        BGT     4$              ;;BRAN H IF YES
(1)  013340  042766  000040  000004        BIC     #40,4(SP)       ;;MAK  IT UPPER CASE
(1)  013346  000002                4$:     RTI                     ;;GO BACK TO USER
(2)                                        ;;*****************************************************************
(1)                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                        ;*CALL:
(1)                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  013350  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  013352  012703  013456        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  013356  022703  013466        2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
(1)  013362  101405                        BLOS    4$              ;;BR IF YES
(1)  013364  104410                        RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  013366  112613                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  013370  122713  000177        10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  013374  001003                        BNE     3$              ;;SKIP IF NOT
(1)  013376  104401  001170        4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  013402  000763                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  013404  111337  013454        3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  013410  104401  013454                TYPE    ,9$
(1)  013414  122723  000015                CMPB    #15,(R3)+       ;;CHEC. FOR RETURN
(1)  013420  001356                        BNE     2$              ;;LOOP IF NOT RETURN
(1)  013422  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  013426  104401  001172                TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  013432  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  013434  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  013436  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  013444  012766  013456  000004        MOV     #$TTYIN,4(SP)
(1)  013452  000002                        RTI                     ;;RETURN
(1)  013454     000                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  013455     000                        .BYTE   0               ;;TERMINATOR
(1)  013456  000010                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  013466  052536  005015     000 $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
(1)  013473     136  006507  000012 $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
(1)  013500  005015  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  013506  020075     000
(1)  013511     040  047040  053505 $MNEW:  .ASCIZ  /  NEW = /
(1)  013516  036440  000040
1467                               .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 (1)
 (2)                               ;;*****************************************************************
```

```
    (1)                                 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
    (1)                                 ;*CHANGE IT TO BINARY.
    (1)                                 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
    (1)                                 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
    (1)                                 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
    (1)                                 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
    (1)                                 ;*CALL:
    (1)                                 ;*      RDOCT                   ;;READ AN OCTAL NUMBER
    (1)                                 ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
    (1)                                 ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
    (1)
    (1)  013522  011646                 $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
    (1)  013524  016666  000004  000002         MOV     4(SP),2(SP)     ;;INPUT NUMBER
    (3)  013532  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
    (3)  013534  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    (3)  013536  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
    (1)  013540  104411                 1$:     RDLIN                   ;;READ AN ASCIZ LINE
    (1)  013542  012600                         MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
    (1)  013544  010037  013650                 MOV     R0,5$           ;;AND SAVE IT
    (1)  013550  005001                         CLR     R1              ;;CLEAR DATA WORD
    (1)  013552  005002                         CLR     R2
    (1)  013554  112046                 2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
    (1)  013556  001420                         BEQ     3$              ;;IF ZERO GET OUT
    (1)  013560  122716  000060                 CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
    (1)  013564  003026                         BGT     4$              ;;IS AN OCTAL DIGIT
    (1)  013566  122716  000067                 CMPB    #'7,(SP)
    (1)  013572  002423                         BLT     4$
    (1)  013574  006301                         ASL     R1              ;;*2
    (1)  013576  006102                         ROL     R2
    (1)  013600  006301                         ASL     R1              ;;*4
    (1)  013602  006102                         ROL     R2
    (1)  013604  006301                         ASL     R1              ;;*8
    (1)  013606  006102                         ROL     R2
    (1)  013610  042716  177770                 BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
    (1)  013614  062601                         ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
    (1)  013616  000756                         BR      2$              ;;LOOP
    (1)  013620  005726                 3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
    (1)  013622  010166  000012                 MOV     R1,12(SP)       ;;SAVE THE RESULT
    (1)  013626  010237  013660                 MOV     R2,$HIOCT
    (3)  013632  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
    (3)  013634  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
    (3)  013636  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
    (1)  013640  000002                         RTI                     ;;RETURN
    (1)  013642  005726                 4$.     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
    (1)  013644  105010                         CLRB    (R0)            ;;SET A TERMINATOR
    (1)  013646  104401                         TYPE                    ;;TYPE UP THRU THE BAD CHAR.
    (1)  013650  000000                 5$:     .WORD   0
    (1)  013652  104401  001170                 TYPE    ,$QUES          ;;"?" "CR" & "LF"
    (1)  013656  000730                         BR      1$              ;;TRY AGAIN
    (1)  013660  000000                 $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
 1468                                   .SBTTL  POWER DOWN AND UP ROUTINES
    (1)
    (2)                                 ;;******************************************************************
    (1)                                 ;POWER DOWN ROUTINE
    (1)  013662  012737  014026  000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
    (1)  013670  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
```

```
(3)  013676  010046                        MOV     R0,-(SP)          ;;PUSH R0 ON STACK
(3)  013700  010146                        MOV     R1,-(SP)          ;;PUSH R1 ON STACK
(3)  013702  010246                        MOV     R2,-(SP)          ;;PUSH R2 ON STACK
(3)  013704  010346                        MOV     R3,-(SP)          ;;PUSH R3 ON STACK
(3)  013706  010446                        MOV     R4,-(SP)          ;;PUSH R4 ON STACK
(3)  013710  010546                        MOV     R5,-(SP)          ;;PUSH R5 ON STACK
(3)  013712  017746  165222                MOV     @SWR,-(SP)        ;;PUSH @SWR ON STACK
(1)  013716  010637  014032                MOV     SP,$SAVR6         ;;SAVE SP
(1)  013722  012737  013734  000024        MOV     #$PWRUP,@#PWRVEC  ;;SET UP VECTOR
(1)  013730  000000                        HALT
(1)  013732  000776                        BR      .-2               ;;HANG UP
(1)
(2)                                 ;;*****************************************************************
(1)                                 ;POWER UP ROUTINE
(1)  013734  012737  014026  000024 $PWRUP: MOV     #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
(1)  013742  013706  014032                MOV     $SAVR6,SP         ;;GET SP
(1)  013746  005037  014032                CLR     $SAVR6            ;;WAIT LOOP FOR THE TTY
(1)  013752  005237  014032        1$:     INC     $SAVR6            ;;WAIT FOR THE INC
(1)  013756  001375                        BNE     1$                ;;OF  WORD
(3)  013760  012677  165154                MOV     (SP)+,@SWR        ;;POP STACK INTO @SWR
(3)  013764  012605                        MOV     (SP)+,R5          ;;POP STACK INTO R5
(3)  013766  012604                        MOV     (SP)+,R4          ;;POP STACK INTO R4
(3)  013770  012603                        MOV     (SP)+,R3          ;;POP STACK INTO R3
(3)  013772  012602                        MOV     (SP)+,R2          ;;POP STACK INTO R2
(3)  013774  012601                        MOV     (SP)+,R1          ;;POP STACK INTO R1
(3)  013776  012600                        MOV     (SP)+,R0          ;;POP STACK INTO R0
(1)  014000  012737  013662  000024        MOV     #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
(1)  014006  012737  000340  000026        MOV     #340,@#PWRVEC+2   ;;PRIO:7
(1)  014014  104401                        TYPE                      ;;REPORT THE POWER FAILURE
(1)  014016  014034                $PWRMG: .WORD   PWRMSG            ;;POWER FAIL MESSAGE POINTER
(1)  014020  012716                        MOV     (PC)+,(SP)        ;;RESTART AT RESTRT
(1)  014022  002406                $PWRAD: .WORD   RESTRT            ;;RESTART ADDRESS
(1)  014024  000002                        RTI
(1)  014026  000000                $ILLUP: HALT                      ;;THE POWER UP SEQUENCE WAS STARTED
(1)  014030  000776                        BR      .-2               ;; BEFORE THE POWER DOWN WAS COMPLETE
(1)  014032  000000                $SAVR6: 0                         ;;PUT THE SP HERE
1469 014034  005015  042522  052123 PWRMSG: .ASCIZ  <15><12>/RESTARTED FROM PWR FAIL/
     014042  051101  042524  020104
     014050  051106  046517  050040
     014056  051127  043040  044501
     014064  000114
1470                                .EVEN
1471                                .SBTTL  TRAP DECODER
(1)
(2)                                 ;;*****************************************************************
(1)                                 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)                                 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)                                 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)                                 ;*GO TO THAT ROUTINE.
(1)
(1)  014066  010046                $TRAP:  MOV     R0,-(SP)          ;;SAVE R0
(1)  014070  016600  000002                MOV     2(SP),R0          ;;GET TRAP ADDRESS
(1)  014074  005740                        TST     -(R0)             ;;BACKUP BY 2
(1)  014076  111000                        MOVB    (R0),R0           ;;GET RIGHT BYTE OF TRAP
(1)  014100  006300                        ASL     R0                ;;POSITION FOR INDEXING
(1)  014102  016000  014122                MOV     $TRPAD(R0),R0     ;;INDEX TO TABLE
```

```
   (1)  014106  000200                        RTS     R0              ;;GO TO ROUTINE
   (1)
   (1)
   (1)                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
   (1)
   (1)  014110  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
   (1)  014112  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
   (1)  014120  000002                          RTI                     ;;RESTORE THE PSW
   (1)
   (3)                                  .SBTTL  TRAP TABLE
   (3)
   (3)                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
   (3)                                  ;*BY THE "TRAP" INSTRUCTION.
   (3)
   (3)                                  ;       ROUTINE
   (3)                                  ;       -------
   (3)  014122  014110                  $TRPAD: .WORD   $TRAP2
   (3)  014124  010630                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
   (3)  014126  011456                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
   (3)  014130  011432                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
   (3)  014132  011472                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
   (3)  014134  011660                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
   (1)
   (3)  014136  013006                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
   (1)
   (3)  014140  012736                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
   (3)  014142  013220                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
   (3)  014144  013350                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
   (3)  014146  013522                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
  1472
  1473                                          .SBTTL  ASCII MESSAGES
  1474  014150  042522  020107  044524  EM1:    .ASCIZ  /REG TIMEOUT ER/
        014156  042515  052517  020124
        014164  051105     000
  1475  014167     122  043505  051040  EM2:    .ASCIZ  'REG READ/WRITE ER'
        014174  040505  027504  051127
        014202  052111  020105  051105
        014210     000
  1476  014211     102  051525  051040  EM3:    .ASCIZ  /BUS RESET ER/
        014216  051505  052105  042440
        014224  000122
  1477  014226  047106  052103  041040  EM4:    .ASCIZ  /FNCT BITS FAILED TO SET STAT BITS/
        014234  052111  020123  040506
        014242  046111  042105  052040
        014250  020117  042523  020124
        014256  052123  052101  041040
        014264  052111  000123
  1478  014270  042522  042101  020131  EM5:    .ASCIZ  /READY INTR FAILURE/
        014276  047111  051124  043040
        014304  044501  052514  042522
        014312     000
  1479  014313     122  040505  054504  EM6:    .ASCIZ  /READY CLR OR SET ER/
        014320  041440  051114  047440
        014326  020122  042523  020124
        014334  051105     000
  1480  014337     123  040524  052524  EM7:    .ASCIZ  /STATUS ER ON XFER/
```

```
                014344  020123  051105  047440
                014352  020116  043130  051105
                014360     000
1481            014361     127  051117  020104  EM10:   .ASCIZ  /WORD COUNT ER ON XFER/
                014366  047503  047125  020124
                014374  051105  047440  020116
                014402  043130  051105     000
1482            014407     102  043125  042506  EM11:   .ASCIZ  /BUFFER ADRS ER ON XFER/
                014414  020122  042101  051522
                014422  042440  020122  047117
                014430  054040  042506  000122
1483            014436  040504  040524  042440  EM12:   .ASCIZ  /DATA ER FROM MEM/
                014444  020122  051106  046517
                014452  046440  046505     000
1484            014457     104  052101  020101  EM13:   .ASCIZ  /DATA ER TO MEM/
                014464  051105  052040  020117
                014472  042515  000115
1485            014476  044523  043516  042514  EM14:   .ASCIZ  /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
                014504  041440  041531  042514
                014512  047440  043106  042040
                014520  042111  047040  052117
                014526  046040  041517  020113
                014534  052517  020124  050103
                014542  000125
1486            014544  044523  043516  042514  EM15:   .ASCIZ  /SINGLE CYCLE ON LOCKED OUT CPU/
                014552  041440  041531  042514
                014560  047440  020116  047514
                014566  045503  042105  047440
                014574  052125  041440  052520
                014602     000
1487            014603     015  050012  042514  WARN:   .ASCII  <15><12>/PLEASE DISABLE ''REV11'' MEMORY REFRESH OPTION/
                014610  051501  020105  044504
                014616  040523  046102  020105
                014624  051042  053105  030461
                014632  020042  042515  047515
                014640  054522  051040  043105
                014646  042522  044123  047440
                014654  052120  047511     116
1488            014661     015  040412  042116          .ASCIZ  <15><12>/AND ENABLE PROCESSOR MEMORY REFRESH   /
                014666  042440  040516  046102
                014674  020105  051120  041517
                014702  051505  047523  020122
                014710  042515  047515  054522
                014716  051040  043105  042522
                014724  044123  020040  000040
1489            014732  042516  020130  047514  EM16:   .ASCIZ  /NEX LOGIC ER/
                014740  044507  020103  051105
                014746     000
1490            014747     103  041531  042514  EM17:   .ASCIZ  /CYCLE FAILED TO CLK DBR  IN)/
                014754  043040  044501  042514
                014762  020104  047524  041440
                014770  045514  042040  051102
                014776  024040  047111  000051
1491            015004  040504  040524  042440  EM20:   .ASCIZ  ''DATA ER FROM I/O PAGE (XCSR)''
                015012  020122  051106  046517
                015020  044440  047457  050040
```

```
                015026  043501  020105  054050
                015034  051503  024522     000
1492            015041     105  051122  041520  DH1:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  EXPCT   RCVD/
                015046  020040  052040  052123
                015054  052516  020115  041040
                015062  051525  042101  020122
                015070  042440  050130  052103
                015076  020040  051040  053103
                015104  000104
1493            015106  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   TSTNUM  BUSADR  ADRS    EXPCT   RCVD/
                015114  020040  051524  047124
                015122  046525  020040  052502
                015130  040523  051104  020040
                015136  042101  051522  020040
                015144  020040  054105  041520
                015152  020124  020040  041522
                015160  042126     000
1494            015163     105  051122  041520  DH3:    .ASCIZ  /ERRPC   TSTNUM  BUSADR/
                015170  020040  052040  052123
                015176  052516  020115  041040
                015204  051525  042101  000122
1495
1496                                             .EVEN
1497            015212  001116  001534  001122  DT1:    $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
                015220  001124  001126  000000
1498            015226  001116  001534  001122  DT2:    $ERRPC,TSTNUM,$BDADR,$GDADR,$GDDAT,$BDDAT,0
                015234  001120  001124  001126
                015242  000000
1499            015244  001116  001534  001122  DT3:    $ERRPC,TSTNUM,$BDADR,0
                015252  000000
1500            015254  005015  051511  052040  KDF11B: .ASCIZ  <15><12>/IS THE PROCESSOR KDF11-B?  /
                015262  042510  050040  047522
                015270  042503  051523  051117
                015276  045440  043104  030461
                015304  041055  020077  000040
1501            015312  005015  051124  047101  IOTEST: .ASCIZ  <15><12>.TRANSFERS FOR I/O PAGE?  .
                015320  043123  051105  020123
                015326  047506  020122  027511
                015334  020117  040520  042507
                015342  020077  000040
1502            015346  005015  042101  051104  IOADR:  .ASCIZ  <15><12>\ADDRESS OF I/O PAGE?  \
                015354  051505  020123  043117
                015362  044440  047457  050040
                015370  043501  037505  020040
                015376     000
1503                    015400                           .EVEN
1504                                             ;;*****************************************************************
1505                                             ;'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
1506                                             ;NPR OPERATIONS - IT IS 200 WORDS LONG
1507                                             ;;*****************************************************************
1508            015400  000000                   DBUF:   0               ;1ST ADRS OF DATA BUFFER
1509                    000001                           .END
```

```
ABASE = 172410            377#     390
ACDW1 = 000000            390
ACDW2 = 000000            390
ACPUOP= 000000            390
ADDW0 = 000000            390
ADDW1 = 000000            390
ADDW10= 000000            390
ADDW11= 000000            390
ADDW12= 000000            390
ADDW13= 000000            390
ADDW14= 000000            390
ADDW15= 000000            390
ADDW2 = 000000            390
ADDW3 = 000000            390
ADDW4 = 000000            390
ADDW5 = 000000            390
ADDW6 = 000000            390
ADDW7 = 000000            390
ADDW8 = 000000            390
ADDW9 = 000000            390
ADEVCT= 000000            390
ADEVM = 000001            379#     390
AENV  = 000000            390
AENVM = 000000            390
AFATAL= 000000            390
ALTERN  007452           1199     1240#
AMADR1= 000000            390
AMADR2= 000000            390
AMADR3= 000000            390
AMADR4= 000000            390
AMAMS1= 000000            390
AMAMS2= 000000            390
AMAMS3= 000000            390
AMAMS4= 000000            390
AMSGAD= 000000            390
AMSGLG= 000000            390
AMSGTY= 000000            390
AMTYP1= 000000            390
AMTYP2= 000000            390
AMTYP3= 000000            390
AMTYP4= 000000            390
APASS = 000000            390
APRIOR= 000000            390
APTCSU= 000040           1451     1452#
APTENV= 000001           1451     1452#    1455
APTSIZ= 000200            514     1452#
APTSPO= 000100           1451     1452#
ASWREG= 000000            390
ATESTN= 000000            390
AUNIT = 000000            390
AUSWR = 000000            390
AVECT1= 000124            378#     390
AVECT2= 000000            390
BIT0  = 000001            376#
BIT00 = 000001            376#     610      615      664      1344
BIT01 = 000002            376#
```

```
BIT02 = 000004            376#
BIT03 = 000010            376#
BIT04 = 000020            376#
BIT05 = 000040            376#
BIT06 = 000100            376#
BIT07 = 000200            376#    539
BIT08 = 000400            376#   1465
BIT09 = 001000            376#   1455    1465
BIT1  = 000002            376#
BIT10 = 002000            376#   1455
BIT11 = 004000            376#   1465
BIT12 = 010000            376#
BIT13 = 020000            376#   1055    1455
BIT14 = 040000            376#   1465
BIT15 = 100000            376#
BIT2  = 000004            376#
BIT3  = 000010            376#
BIT4  = 000020            376#
BIT5  = 000040            376#
BIT6  = 000100            376#    746
BIT7  = 000200            376#
BIT8  = 000400            376#
BIT9  = 001000            376#
BPTVEC= 000014            376#
CKDAT   010460            959    1405#
CKDAT1  010526           1061    1152    1190    1424#
CKSTAT  010156            830     873     915     949    1043    1142    1180    1216    1258    1326#
CKSWR = 104407           1455    1462    1465    1471#
CORSZ   001540            507#    540     551*    1162
CORSZR  002304            541     546#
CR    = 000015            376#   1451
CRLF  = 000200            376#    528    1451
DBUF    015400            508     553     818     819*    839     862     883     885     904     924     937     970     999
                         1031    1127    1195    1233    1275    1359    1363    1368    1405    1412    1508#
DBUFP   001542            508#    553*   1160*   1161*   1162    1165    1195*   1203*   1233*   1245*   1275*   1342    1382
                         1425
DDISP = 177570            376#    390     514
DH1     015041            393     399     405     411     417     423     429     435     441     471     477    1492#
DH2     015106            447     453     486    1493#
DH3     015163            459     465    1494#
DISPLA  001142            390#    514*   1455*   1465*
DISPRE  000174            382#    514
DMAP    001536            506#    555*    556*    558    1283*   1292
DRVBAR  001522            494#    605     608*    609     656*    663     666     780     782*    783     788     818*    862*
                          904*    937*    970*    999*   1031*   1072*   1098    1102    1127*   1165*   1204*   1246*   1286*
                         1343    1347
DRVCSR  001524            495#    678     682*    683     693     699     701*    703     709     724     727*    728     74'
                          746*    748     752     756     764     766*    767     771*    773     779*    787*    792*    798*
                          804*    820*    821*    823     825     826*    864*    865*    867     869     906*    907*    908
                          910     911*    940*    941*    942     944     945*    964     972*    973*    980     981*    993
                         1001*   1002*   1007    1008*   1017    1034*   1035*   1036    1038    1039*   1074*   1075*   1076
                         1078    1079*   1083    1087    1107*   1108    1112    1116    1120    1130*   1131*   1135    1137
                         1138*   1168*   1169*   1173    1175    1176*   1207*   1208*   1209    1211    1212*   1249*   1250*
                         1251    1253    1254*   1287*   1314*   1326    1327*   1331
DRVCT0  001530            500#    521     740*    745*   1289*   1306*   1315*
DRVCT2  001532            501#    526    1290*   1307*   1315    1316*
```

```
DRVDBR  001526          496#    519     625     628*    629     641     657*    668     670     795     797*    799*    800
                        806     841     844     863*    888     926     929     938*    1206*   1227    1230    1248*   1269
                       1272    1288*   1408    1439
DRVWCR  001520          493#    515     566     574     588     591*    592     655*    659     661     817*    861*    903*
                        936*    969*    998*    1032*   1033*   1071*   1091    1095    1128*   1166*   1202*   1244*   1285*
                       1334    1336
DSWR   = 177570         376#    390     514
DT1     015212          394     400     406     412     418     424     430     456     442     472     478     1497#
DT2     015226          448     454     487     1498#
DT3     015244          460     466     1499#
EMTVEC= 000030          376#    514*
EM1     014150          392     1474#
EM10    014361          434     1481#
EM11    014407          440     1482#
EM12    014436          446     1483#
EM13    014457          452     1484#
EM14    014476          458     1485#
EM15    014544          464     1486#
EM16    014732          470     1489#
EM17    014747          476     1490#
EM2     014167          398     1475#
EM20    015004          485     1491#
EM3     014211          404     1476#
EM4     014226          410     1477#
EM5     014270          416     1478#
EM6     014313          422     1479#
EM7     014337          428     1480#
ERRVEC= 000004          376#    514*    547*    552*    573*    584*    1465*
GNS    = ****** U       382     528     1471
GTSWR  = 104406         528     1471#
HT     = 000011         376#    1451
IOADR   015346          1502#
IOPAGE  001544          509#    1240    1245    1246    1268    1273
IOTEST  015312          1501#
IOTVEC= 000020          376#    514*
KDF     001546          510#    535*    536     537
KDF11B  015254          533     1500#
LDBUF   010336          902     1359#
LDBUF1  010410          1027    1126    1164    1382#
LF     = 000012         376#    1451
MTPS   = 106427         380#
NXDEV   007672          1235    1241    1281#
NXDEV1  007674          560     1282#   1293
PIRQ   = 177772         376#
PIRQVE= 000240          376#
PR0    = 000000         376#
PR1    = 000040         376#
PR2    = 000100         376#
PR3    = 000140         376#
PR4    = 000200         376#
PR5    = 000240         376#
PR6    = 000300         376#
PR7    = 000340         376#
PS     = 177776         376#
PSW    = 177776         376#
PWRMSG  014034          1468    1469#
```

```
PWRVEC= 000024   376#   514*   1468*
RDCHR = 104410   534    1466   1471#
RDLIN = 104411   1467   1471#
RDOCT = 104412   1471#
RESTRT  002406   557    559    561#   1294   1468
RESVEC= 000010   376#
RSTVEC  010132   696    760    853    897    931    961    990    1022   1065   1122   1156   1196   1234
                 1276   1314#
SETUP2  002022   517#   520
SETUP3  002052   524#   527
SETVEC  010112   675    813    857    900    934    965    994    1025   1069   1124   1158   1200   1242
                 1305#
STACK = 001100   376#   514    562
START   001550   382    514#
START1  002012   515#   1295
STKLMT= 177774   376#
SWR     001140   390#   514*   528    539*   1198   1455   1465   1466*  1468*
SWRCK   012274   1455   1460#
SWREG   000176   382#   514    528    1466
SW0   = 000001   376#
SW00  = 000001   376#
SW01  = 000002   376#
SW02  = 000004   376#
SW03  = 000010   376#
SW04  = 000020   376#
SW05  = 000040   376#
SW06  = 000100   376#
SW07  = 000200   376#
SW08  = 000400   376#
SW09  = 001000   376#
SW1   = 000002   376#
SW10  = 002000   376#
SW11  = 004000   376#
SW12  = 010000   376#
SW13  = 020000   376#
SW14  = 040000   376#
SW15  = 100000   376#
SW2   = 000004   376#
SW3   = 000010   376#
SW4   = 000020   376#
SW5   = 000040   376#
SW6   = 000100   376#
SW7   = 000200   376#
SW8   = 000400   376#
SW9   = 001000   376#
TBITVE= 000014   376#
TKVEC = 000060   376#
TMAIN   000000   381#
TPVEC = 000064   376#
TRAPVE= 000034   376#   514*
TRTVEC= 000014   376#
TSTNUM  001534   505#   1460*  1497   1498   1499
TST1    002442   568#
TST10   003442   698#
TST11   003516   705    708#
TST12   003600   722#
```

```
TST13   003676            737#
TST14   004036            762#
TST15   004140            775     778#
TST16   004242            794#
TST17   004356            808     811#
TST2    002552            586#
TST20   004606            855#
TST21   005030            899#
TST22   005236            933#
TST23   005412            963#
TST24   005566            992#
TST25   005770           1024#
TST26   006226           1067#
TST27   006606           1123#
TST3    002650            603#
TST30   007000           1157#
TST31   007224           1197#
TST4    002766            619     623#
TST5    003064            640#
TST6    003144            650     653#
TST7    003272            669     673#
TYPDS = 104405           1295    1471#
TYPE  = 104401            528     533     536     542    1295    1451    1453    1454    1455    1464    1466    1467    1468
                         1471#
TYPOC = 104402           1464    1466    1471#
TYPON = 104404           1471#
TYPOS = 104403           1471#
WARN    014603            542    1487#
$APTHD  001000            389#
$ASTAT= ****** U         1452
$ATYC   011210           1452#
$ATY1   011164           1452#
$ATY3   011172           1451    1452#
$ATY4   011202           1452#   1455
$AUTOP  001134            390#    528*    531    1466
$BASE   001250            390#    516
$BDADR  001122            390#    576*    588*    605*    625*    642*    661*    666*    670*    678*    699*    710*    724*
                          741*    764*    780*    795*    823*    844*    867*    888*    908*    929*    942*    964*    993*
                         1036*   1076*   1087*   1095*   1102*   1112*   1120*   1135*   1173*   1209*   1230*   1251*   1272*
                         1331*   1336*   1347*   1408*   1439*   1497    1498    1499
$BDDAT  001126            390#    572*    592*    593     609*    610*    611     629*    630     645*    646     647     648*
                          649     659*    663*    664*    668*    683*    684*    685     693*    694*    703*    704     713*
                          714     715     716*    717     728*    729*    730     748*    752*    756*    757     767*    768
                          773*    774     783*    784     788*    800*    801     806*    807     825*    841*    842     869*
                          885*    886     910*    926*    927     944*    968*    977*    980*    987     997*   1004*   1007*
                         1014    1017*   1019    1038*   1078*   1083*   1085    1091*   1093    1098*   1100    1108*   1110
                         1116*   1118    1132*   1133*   1137*   1170*   1171*   1175*   1211*   1227*   1228    1253*   1269*
                         1270    1326*   1329    1334*   1343*   1344*   1345    1409*   1436*   1497    1498
$BELL   001104            390#    1455
$CDW1   001254            390#
$CDW2   001256            390#
$CHARC  011160           1451#*
$CKSWR  012736           1466#    1471
$CMTAG  001100            390#     514
$CM3  = 000000            390#
$CNTLG  013473           1466#
```

```
$CNTLU  013466          1466#
$CPUOP  001222           390#
$CRLF   001171           390#   1451    1455    1464    1466    1467
$DBLK   012074          1454#
$DDW0   001260           390#    1460*
$DDW1   001262           390#
$DDW10  001304           390#
$DDW11  001306           390#
$DDW12  001310           390#
$DDW13  001312           390#
$DDW14  001314           390#
$DDW15  001316           390#
$DDW2   001264           390#
$DDW3   001266           390#
$DDW4   001270           390#
$DDW5   001272           390#
$DDW6   001274           390#
$DDW7   001276           390#
$DDW8   001300           390#
$DDW9   001302           390#
$DEVCT  001204           390#    563*
$DEVM   001252           390#    555
$DOAGN  010065          1295#
$DTBL   012064          1454#
$ENDAD  010056           387    1295#
$ENDCT  010024          1295#
$ENDMG  010075          1295#
$ENULL  010072          1295#
$ENV    001214           390#    528    1451    1452    1455
$ENVM   001215           390#    514    1451    1452
$EOP    007770          1284    1295#
$EOPCT  010016          1295#
$ERFLG  001103           390#   1455*   1465*
$ERMAX  001115           390#    514*   1465*
$ERROP  012104           514    1455#
$ERRPC  001116           390#   1455*   1464    1497    1498    1499
$ERRTB  001320           390#   1464
$ERRTY  012312          1461    1464#
$ERTTL  001112           390#   1455*
$ESCAP  001162           390#    514*   1455    1465*
$ETABL  001214           390#
$ETEND  001320           389     390#
$FATAL  001176           390#   1452*
$FFLG   011430          1452#*
$FILLC  001156           390#   1451
$FILLS  001155           390#   1451
$GDADR  001120           390#    839*   883*   924*   1231*  1273*  1410*  1437*  1498
$GDDAT  001124           390#    571*   590*   591    593    596*   599*   600*   607*   608    611    614*   615*
                         618*    620*   627*   628    630    633*   636*   637*   644*   649    654*   680*   681*
                         685     692*   700*   704    712*   717    725*   730    733*   742*   757    765*   768
                         772*    774    781*   784    790*   796*   797    801    805*   807    824*   840*   842
                         868*    884*   886    909*   925*   927    943*   979*   1006*  1018*  1019   1037*  1077*
                        1084*   1085   1092*  1093   1099*  1100   1109*  1110   1117*  1118   1136*  1174*  1210*
                        1226*   1228   1252*  1268*  1270   1328*  1329   1337*  1340*  1341*  1342*  1345   1438*
                        1497    1498
$GET42  010046          1295#
```

```
$GTSWR  013006          1466#   1471
$HD     = 000003          371
$HIBTS  001000          389#
$HIOCT  013660          1467#*
$ICNT   001104          390#    1465*
$ILLUP  014026          1468#
$INTAG  001135          390#    1466
$ITEMB  001114          390#    1455*   1464
$LF     001172          390#    1451    1455    1466    1467
$LFLG   011427          1452#*
$LPADR  001106          390#     514*    568*   1465*
$LPERR  001110          390#     514*    570*    587*    604*    624*    677*    723*    812*    856*    1455    1465*
$MADR1  001226          390#
$MADR2  001232          390#
$MADR3  001236          390#
$MADR4  001242          390#
$MAIL   001174          389     390#     514     528     568    1451    1455    1465
$MAMS1  001224          390#
$MAMS2  001230          390#
$MAMS3  001234          390#
$MAMS4  001240          390#
$MBADR  001002          389#
$MFLG   011426          1452#*
$MNEW   013511          1466#
$MSGAD  001210          390#    1452*
$MSGLG  001212          390#    1452*
$MSGTY  001174          390#    1452*
$MSWR   013500          1466#
$MTYP1  001225          390#
$MTYP2  001231          390#
$MTYP3  001235          390#
$MTYP4  001241          390#
$MXCNT  012734          1465#
$NULL   001154          390#    1451
$NWTST= 000001          568#     586#    603#    623#    640#    653#    673#    698#    708#    722#    737#    762#    778#
                        794#     811#    855#    899#    933#    963#    992#   1024#   1067#   1123#   1157#   1197#
$OCNT   011654          1453#*
$OMODE  011656          1453#*
$OVER   012720          1465#
$PASS   001202          390#     514*   1295*   1465
$PASTM  001006          389#
$PURAD  014022          1468#
$PURDN  013662          514     1468#
$PURMG  014016          1468#
$PURUP  013734          1468#
$QUES   001170          390#    1451    1455    1466    1467
$RDCHR  013220          1466#   1471
$RDDEC= ****** U        1471
$RDLIN  013350          1466#   1471
$RDOCT  013522          1466#   1471
$RDSZ = 000010          1466#
$RTNAD  010070          1295#
$R2A  = ****** U        1471
$SAVRE- ****** U        1471
$SAVR6  014032          1468#*
$SCOPE  012446           514    1465#
```

```
$SETUP= 000117          512#   514    528    1295   1455   1465   1466
$STUP = 177777          512#
$SVLAD  012664         1465#
$SVPC = 000106          387#
$SWR  = 167400          371#   372#   375    390    514    568    586    603    623    640    653    673    698
                        708    722    737    762    778    794    811    855    899    933    963    992    1024
                       1067   1123   1157   1197   1295   1455   1465   1468
$SWREG  001216          390#   514
$SWRMK= 000300          373#   375    1465
$TESTN  001200          390#   568*   1465*
$TIMES  001160          390#   514*   653*   673*   698*   737*   899*   933*   963*   992*   1024*  1067*  1123*
                       1157*  1295*  1465*
$TKB    001146          390#  1451    1466
$TKS    001144          390#  1451    1466*
$TN   - 000032          371#   374#   568#   586#   603#   619    623#   640#   650    653#   669    673#   698#
                        705    708#   722#   737#   762#   775    778#   794#   808    811#   855#   899#   933#
                        963#   992#  1024#  1067#  1123#  1157#  1197#
$TPB    001152          390#  1451*
$TPFLG  001157          390#  1451
$TPS    001150          390#  1203    1204   1226   1231   1451
$TRAP   014066          514   1471#
$TRAP2  014110         1471#
$TRP  = 000013         1471#
$TRPAD  014122         1471#
$TSTM   001004          389#
$TSTNM  001102          390#   569*   1295*  1455   1460   1465*
$TTYIN  013456         1466#
$TYPBN= ****** U        1471
$TYPDS  011660         1454#  1471
$TYPE   010630         1451#  1452    1471
$TYPEC  011042         1451#  1466
$TYPEX  011162         1451#
$TYPOC  011456         1453#  1471
$TYPON  011472         1453#  1471
$TYPOS  011432         1453#  1471
$UNIT   001206          390#   554*    563    564   1291*
$UNITM  001010          389#
$USWR   001220          390#
$VECT1  001244          390#   522
$VECT2  001246          390#
$XOFF = 000023         1451
$XON  = 000021         1451    1466
$XTSTR  012462         1465#
$$GET4= 000000         1295#
$OFILI  011655         1453#*
$40CAT= ****** U        1455   1465
.     = 015402          382#   383#   387#   388#   389#   390#   514    528#   1295   1451   1452#  1454#  1455
                       1464#  1465   1466#  1467   1468   1503#
.$ASTA= ****** U        1452
.$X     001000          389#

. ABS.  015402     000

 ERRORS DETECTED:  0
```

```
CVDRAB,CVDRAB/CRF:SYM/NL:TOC=CVDRAB.P11
RUN-TIME:  19 9 .9 SECONDS
RUN-TIME RATIO:  116/29=3.8
CORE USED:  27K  (54 PAGES)
```