DRV11B

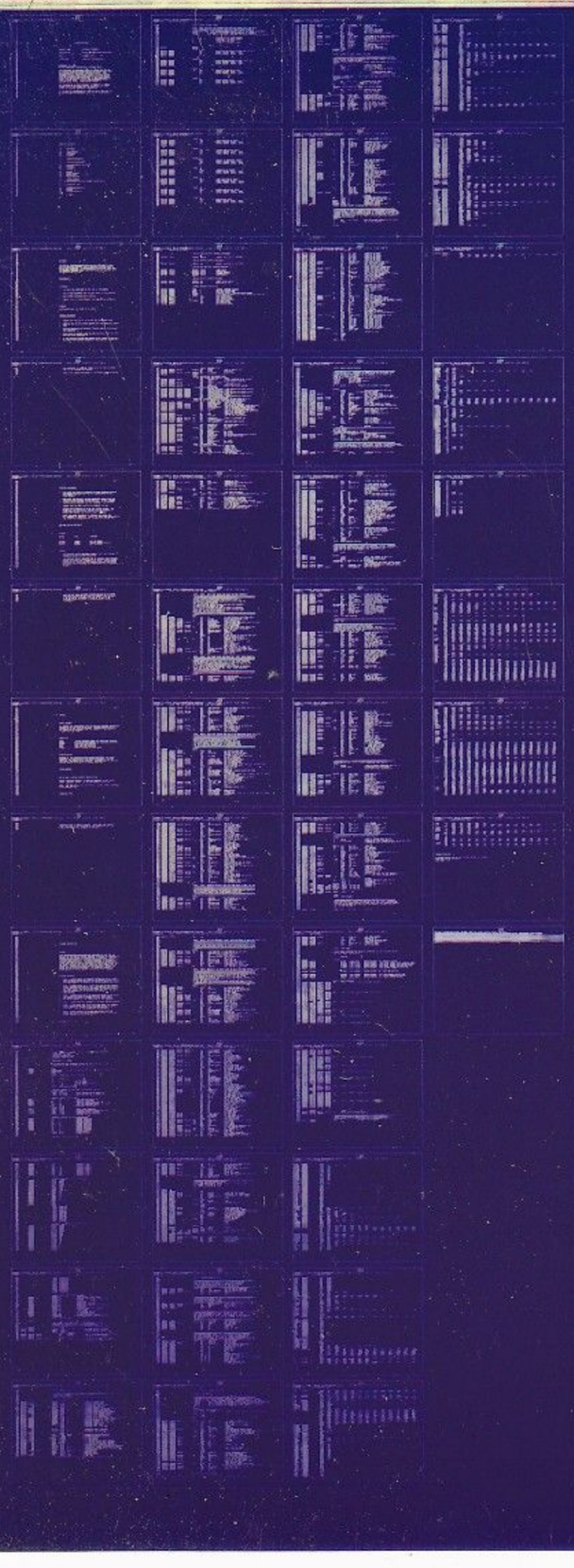
INTERPROCESSOR EXERCISER

MD-11-DVDRB-A

EP-DVDRB-A-DL-A NOV 1976

COPYRIGHT © 1976

FICHE 1 OF 1 MADE IN U.S.A



```
                        .REM    %
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DVDRB-A
PRODUCT NAME: DRV11B INTERPROCESSOR EXERCISER
DATE: OCTOBER 1976
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1976
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PRO-VIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COM-MITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

1.0 ABSTRACT

THE DRV11B INTERPROCESSOR EXERCISER WAS DESIGNED TO PASS DATA TO AND FROM ANOTHER LSI-11 COMPUTER EQUIPPED WITH A DRV11B DMA INTERFACE. SYNCHRONIZATION IS ACCOMPLISHED THRU TWO SEPERATE PROGRAM START LOCATIONS AND IS MAINTAINED UNTIL INTERRUPTED BY THE USER.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. TWO PDP11/03 COMPUTERS OR TWO LSI-11 PROCESSORS
2. EACH SYSTEM EQUIPPED WITH A DLV11 AND I/O TYPE TERMINAL
3. EACH SYSTEM EQUIPPED WITH A DRV11B
4. CABLES (TWO) TO INTER CONNECT THE TWO DRV11B INTERFACES

2.2 STORAGE

THE PROGRAM USES THE LOWER 2K OF MEMORY

3.0 LOADING 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 READ AND TYPE CHARACTER 'P'.
5. IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STOPS 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).

                            6.  REPEAT THE LOADING PROCEDURE IN THE OTHER COMPUTER SYSTEM.
```

4.0 STARTING PROCEDURE

1. MAKE SURE EACH DRV11B INTERFACE OF BOTH COMPUTER SYSTEMS ARE CABLED TOGETHER THRU THE I/O CONNECTIONS ON THE M7950 MODULE.
2. MAKE SURE THE DEVICE & 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. DEFINE WHAT COMPUTER IS TO BE THE INITIAL SLAVE - THE OTHER WILL BE THE INITIAL MASTER (THE ONE STARTED AS MASTER WILL REPORT THE 'END OF PASS' MESSAGE).
5. START THE SLAVE **FIRST** AT ADDRESS 204 USING THE ODT 'ADDRESS G' SEQUENCE. AFTER THE SLAVE HAS BEEN STARTED THEN START THE OTHER COMPUTER (MASTER) AT ADDRESS 200.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
BIT15=1	100000	HALT ON ERROR
BIT13=1	020000	INHIBIT ERROR TYPEOUTS
BIT10=1	002000	BELL ON ERROR

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 TYPEING 'P'
                                    (CONTINUE).
```

6.0 ERRORS

6.1 ERROR COMMENT

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

6.2 ERROR DATA

ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
BUSADR	DRV11B BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED
MEMADR	MEMORY ADDRESS OF DATA ERROR

6.3 ERROR RECOVERY

BECAUSE OF THE SYNCHRONIZATION ESTABLISHED WITH THE OTHER COMPUTER, ALL ERROR CONDITIONS FORCE A AUTOMATIC LOOP ON TEST UNTIL THE ERROR IS ELIMINATED. SOFTWARE SWITCH REGISTER CONTROL SHOULD NOT AFFECT THE ESTABLISHED SYNC.

7.0 MISCELLANEOUS

7.1 DRV11B BUS & VECTOR ADDRESS MODIFICATION

MODIFY LOCATION 'DRVADR' IF BASE BUS ADDRESS IS NOT 172410
MODIFY LOCATION 'DRVECT' IF VECTOR ADDRESS IS NOT 124

NOTE: USE THE LSI-11 ODT FACILITIES TO MODIFY THESE LOCATIONS AFTER PROGRAM LOAD.

8.0 EXECUTION TIME

```
                                EXECUTION TIME IS ABOUT 1 MINUTE.  NOTE THAT THE 'END OF
                                PASS' MESSAGE IS ONLY REPORTED AT THE COMPUTER WHICH WAS
                                STARTED AS THE INITIAL MASTER (START 200).
```

9.0 PROGRAM DESCRIPTION

9.1 GENERAL

THIS INTERPROCESSOR EXERCISER WAS DESIGNED TO TEST THE I/O ABILITY OF THE DRV11B GENERAL PURPOSE INTERFACE TO COMMUNICATE TO ANOTHER DRV11B LOCATED IN ANOTHER LSI-11 COMPUTER. THE TWO COMPUTERS ARE STARTED AT DIFFERENT ADDRESSES TO ESTABLISH INITIAL SYNCHRONIZATION. THE SLAVE COMPUTER IS STARTED 1ST (START 204) AND THE MASTER COMPUTER IS STARTED 2ND (START 200). THE TERMS 'MASTER' AND 'SLAVE' SHOULD BE USED LOOSELY AS THE MASTER WILL BECOME THE SLAVE AND THE SLAVE WILL BECOME THE MASTER AS THE PROGRAM ADVANCES. THE COMPUTER STARTED AT ADDRESS 200 WILL ALWAYS REPORT THE 'END OF PASS' MESSAGE.

9.2 PROGRAM SEGMENTS

1. MTST1 - MASTER SENDS OUT PROGRAM CONTROLLED SINGLE WORDS THRU THE DATA BUFFER REGISTER AND EXPECTS THE SLAVE TO ECHO EACH WORD BACK TO THE MASTER VIA THE DATA BUFFER REGISTER.

2. MTST2 - MASTER SENDS OUT A 'FNCT' BIT CODE IN THE COMMAND STATUS REGISTER AND EXPECTS THE SLAVE TO ECHO EACH CODE IN ITS 'FNCT' BITS. THE MASTER WILL READ THE 'STAT' BIT CODE FROM THE COMMAND/STATUS REGISTER AND COMPARE IT TO THE CODE WRITTEN.

3. MTST3 - MASTER SENDS OUT A 32 WORD DATA BLOCK TO THE SLAVE AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT, AND BUFFER ADDRESS AT THE COMPLETION OF THE TRANSFER.

4. STST1 - SLAVE ECHOS ALL CHANGES IN THE DATA BUFFER REGISTER

5. STST2 - SLAVE READS THE 'STAT' BIT CODE FROM IT'S COMMAND/ STATUS REGISTER, CONVERTS THIS CODE AND WRITES IT INTO IT'S 'FNCT' BITS IN THE COMMAND/STATUS REGISTER.

6. STST3 - SLAVE RECEIVES A 32 WORD DATA BLOCK FROM THE MASTER AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT, BUFFER ADDRESS, AND DATA CONTENT.

10.0 LISTING

```
                                        ;
                                .TITLE  MAINDEC-11-DVDRB-A  DRV11B INTERPROCESSOR EXERCISER
                                ;*COPYRIGHT (C) 1976
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY R. MOORE
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C0),MAR 21, 1976.
                                ;*
                                $TN=1
000001                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
160000                          $SWR=122000
122000                          $TN=1
000001                          .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          ------------------
                                ;*        15            HALT ON ERROR
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        10            BELL ON ERROR
                                .SBTTL  BASIC DEFINITIONS

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

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

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

                                ;*PRIORITY LEVEL DEFINITIONS
```

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

```
                                .EQUIV  BIT09,BIT9
                                .EQUIV  BIT08,BIT8
                                .EQUIV  BIT07,BIT7
                                .EQUIV  BIT06,BIT6
                                .EQUIV  BIT05,BIT5
                                .EQUIV  BIT04,BIT4
                                .EQUIV  BIT03,BIT3
                                .EQUIV  BIT02,BIT2
                                .EQUIV  BIT01,BIT1
                                .EQUIV  BIT00,BIT0

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

                                        .=0
000000                          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                        .=174
000174  000000                  DISPREG:  .WORD  0              ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:    .WORD  0              ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001364                  JMP     @#START1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001372                  JMP     @#START2        ;GO START AS SLAVE COMPUTER
                                        .=100
000100  000104  000200  000002          .WORD   104,200,2       ;IF 'B EVENT' ON Q-BUS IS CONNECTED
                                                                ;JUST DO A RTI (IGNORE IT)
```

```
                                .SBTTL  COMMON TAGS

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

                                        .=1100
001100                  SCMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          $SWR:   .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          $DISPLAY: .WORD DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001164  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001165  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001166  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                        ;;*****************************************************************
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


001170                          SERRTB:
                                ;ERROR  1
001170  006056                          EM1             ;SLAVE DRV11B FAILED TO ECHO DBR CONTENTS
001172  006503                          DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001174  006576                          DT1             ;SERRPC SBDADR  SGDDAT  SBDDAT
001176  000000                          0

                                ;ERROR  2
001200  006120                          EM2             ;SLAVE DRV11B FAILED TO ECHO 'STAT' BITS
001202  006503                          DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001204  006576                          DT1             ;SERRPC SBDADR  SGDDAT  SBDDAT
001206  000000                          0

                                ;ERROR  3
001210  006161                          EM3             ;FAILED TO INTR ON A 'DATI'
001212  006503                          DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001214  006576                          DT1             ;SERRPC SBDADR  SGDDAT  SBDDAT
001216  000000                          0

                                ;ERROR  4
001220  006214                          EM4             ;STATUS ER ON 'DATI'
001222  006503                          DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001224  006576                          DT1             ;SERRPC SBDADR  SGDDAT  SBDDAT
001226  000000                          0
```

```
001230  006240          ;ERROR  5
                        EM5     ;WORD COUNT ER ON 'DATI'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001236  000000          0

001240  006270          ;ERROR  6
                        EM6     ;BUFFER ADRS ER ON 'DATI'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001246  000000          0

001250  006321          ;ERROR  7
                        EM7     ;FAILED TO INTR ON A 'DATO'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001256  000000          0

001260  006354          ;ERROR  10
                        EM10    ;STATUS ER ON 'DATO'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001266  000000          0

001270  006400          ;ERROR  11
                        EM11    ;WORD COUNT ER ON 'DATO'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001276  000000          0

001300  006430          ;ERROR  12
                        EM12    ;BUFFER ADRS ER ON 'DATO'
                        DH1     ;ERRPC  BUSADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001306  000000          0

001310  006461          ;ERROR  13
                        EM13    ;DATA ER ON 'DATO'
                        DH2     ;ERRPC  MEMADR  EXPCT   RCVD
                        DT1     ;SERRPC SBDADR  SGDDAT  SBDDAT
001316  000000          0
```

```
                                ;DRV11B BASE REGISTER ADDRESS ASSIGNMENT
001320  172410                  DRVADR: 172410          ;MODIFY THIS LOC IF DIFFERENT

                                ;DRV11B VECTOR ADDRESS ASSIGNMENT

001322  000124                  DRVECT: 124             ;MODIFY THIS LOC IF DIFFERENT

                                ;DRV11B BUS REGISTER ADDRESS POINTERS

001324  172410                  DRVWCR: 172410          ;WORD COUNT
001326  172412                  DRVBAR: 172412          ;BUFFER ADDRESS
001330  172414                  DRVCSR: 172414          ;COMMAND/STATUS
001332  172416                  DRVDBR: 172416          ;DATA BUFFER

                                ;DRV11B VECTOR ADDRESS POINTERS

001334  000124                  DRVCT0: 124             ;READY & NEX VECTOR
001336  000126                  DRVCT2: 126             ;NEW PSW ON INTR

                                ;COMMON PROGRAM LOCATION(S)

001340  000000                  TIME:   0               ;GENERAL PURPOSE COUNTER
001342  000000                  SAVE:   0               ;REG DATA SAVED HERE
001344  000000                  MSTER:  0               ;0=MASTER START - NON-ZERO=SLAVE START
001346  000001                  ICOUNT: 1               ;# OF TIMES TO REPEAT ALL TESTS BEFORE END PASS MSG
001350  177740                  XPRAM:  -32.            ;XMIT WORD COUNT
001352  006610                          DBUF            ;XMIT BUFFER ADRS
001354  000111                          111             ;XMIT STATUS/CONTROL
001356  177740                  RPRAM:  -32.            ;RCV WORD COUNT
001360  006610                          DBUF            ;RCV BUFFER ADRS
001362  000113                          113             ;RCV STATUS/CONTROL
```

```
                                                .SBTTL  PROGRAM START
001364  005037  001344          START1: CLR     @STER           ;THIS IT MASTER START
001370  000403                          BR      START           ;SKIP NEXT
001372  012737  177777  001344  START2: MOV     #-1,@STER       ;SLAVE START
001400                          START:
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
001400  012706  001100                  MOV     #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001404  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001406  022706  001140                  CMP     #SWR,R6 ;;DONE?
001412  001374                          BNE     .-6             ;;LOOP BACK IF NO
001414  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001420  012737  005144  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001426  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001434  012737  001464  000030          MOV     #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001442  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001450  012737  005012  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001456  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
001464  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001470  012737  001524  000004          MOV     #65$,@#ERRVEC   ;;SET UP ERROR VECTOR
001476  012737  177570  001140          MOV     #SWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
001504  012737  177570  001142          MOV     #DISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
001512  022777  177777  177420          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001520  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001522  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
001524  012716  001532          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001530  000002                          RTI
001532  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001540  012737  000174  001142          MOV     #DISPREG,DISPLAY
001546  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001552  012700  001324                  MOV     #DRVMCR,R0      ;SET UP REG ADRS POINTERS
001556  013701  001320                  MOV     DRVADR,R1       ;GET BASE ADRS
001562  010120                  SETUP2: MOV     R1,(R0)+        ;LOAD EN
001564  062701  000002                  ADD     #2,R1           ;
001570  022700  001334                  CMP     #DRVDBR+2,R0    ;ALL DONE?
001574  001372                          BNE     SETUP2          ;BR IF NOT
001576  012700  001324                  MOV     #DRVCTO,R0      ;SET UP DRV11B VECTOR ADRS POINTER
001602  013701  001322                  MOV     DRVECT,R1       ;GET BASE VECTOR ADRS
001606  010120                  SETUP3: MOV     R1,(R0)+        ;
001610  062701  000002                  ADD     #2,R1           ;
001614  022700  001340                  CMP     #DRVCT2+2,R0    ;ALL DONE?
001620  001372                          BNE     SETUP3          ;BR IF NOT
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001622  005227  177777                  INC     #-1             ;;FIRST TIME?
001626  001051                          BNE     64$             ;;BRANCH IF NO
001630  104400  001666                  TYPE    ,65$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001634  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
```

F02


```
  670  001640  001006                        BNE     66$                     ;;BRANCH IF YES
  671  001642  023727  001140  000176        CMP     SWR,#SWREG              ;;SOFTWARE SWITCH REG SELECTED?
  672  001650  001005                        BNE     67$                     ;;BRANCH IF NO
  673  001652  104405                        GTSWR                           ;;GET SOFT-SWR SETTINGS
  674  001654  000403                        BR      67$
  675  001656  112737  000001  001134 66$:   MOVB    #1,$AUTOB               ;;SET AUTO-MODE INDICATOR
  676  001664                          67$:
  677  001664  000432                        BR      64$                     ;;GET OVER THE ASCIZ
  678                                  ;;65$:  .ASCIZ  <CRLF>#MD-11-DVDRB-A  DRV11B INTERPROCESSOR EXERCISER  #<CRLF>
  679  001752                          64$:
  680  001752  106427  000200                MTPS,   200                     ;SET PRIORITY TO HIGHEST LEVEL
  681  001756  012706  001100                MOV     #STACK,SP               ;ALWAYS RESET STACK PTR
  682  001762  000005                        RESET                           ;INITIALIZE DRV11B BEFORE TESTING
  683  001764  012737  000001  001346        MOV     #1,ICOUNT               ;1ST TIME DO ALL TEST ONCE - THEN 20000(8)
  684  001772  005737  001344                TST     MSTER                   ;START AS MASTER?
  685  001776  001402                        BEQ     1$                      ;YES - GO SEND TO SLAVE & CHECK ECHO
  686  002000  000137  002672                JMP     STST1                   ;NO - GO ECHO MASTER'S DATA
  687  002004  012777  000400  177316  1$:   MOV     #400,@DRVCSR            ;INSURE THAT CYCLE IS SET
```

```
                                .SBTTL  MTST1  TEST THAT SLAVE CAN ECHO THE DBR CORRECTLY
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;MASTER COMPUTER STARTS HERE FROM PROGRAM START
                                ;IT SENDS OUT SINGLE WORDS (FLOATING I/O PTRN) THRU THE
                                ;DBR TO THE SLAVE COMPUTER
                                ;IT LOOKS FOR THE SLAVE TO ECHO THE DBR WORD CORRECTLY
                                ;WITHIN A CERTAIN AMOUNT OF TIME
                                ;IF IT FAILS TO RETURN THE WORD AN ERROR IS REPORTED
                                ;THIS TEST IS NOT EXITED UNTIL ALL DATA PATTERNS HAVE
                                ;BEEN SENT AND RECEIVED CORRECTLY
                                ;;*****************************************************************
                                ;;*****************************************************************
002012                          MTST1:
002012  005001                  1$:     CLR     R1              ;R1 SAYS FLOAT 0 LEFT WHEN ZERO
002014  012700  177776          2$:     MOV     #-2,R0          ;START WITH #177776 IN R0
002020  010077  177306                  MOV     R0,@DRVDBR      ;SEND PATTERN OUT
002024  042777  000400  177276          BIC     #400,@DRVCSR    ;CLRING CYCLE HERE SETS CYCLE IN SLAVE
002032  010037  001124                  MOV     R0,$GDDAT       ;SAVE IN EXPECTED
002036  013737  001332  001122          MOV     DRVDBR,$BDADR   ;SET UP DBR ADRS
002044  032777  000400  177256  3$:     BIT     #400,@DRVCSR    ;CYCLE SETS WHEN SLAVE ECHOS DATA
002052  001774                          BEQ     3$              ;WAIT FOR SLAVE TO RESPOND
002054  017737  177252  001126          MOV     @DRVDBR,$BDDAT  ;READ IT BACK
002062  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
002070  001402                          BEQ     4$              ;BR IF SO
002072  104001                          ERROR   1               ;SLAVE FAILED TO ECHO THE DBR WORD
002074  000751                          BR      2$              ;LOOP ON ERROR ALWAYS
002076  005100                  4$:     COM     R0              ;CONVERT PTRN TO FLOATING 1
002100  005101                          COM     R1              ;TIME TO FLOAT LEFT?
002102  001346                          BNE     2$              ;BR IF NOT
002104  006300                          ASL     R0              ;YES - FLOAT LEFT
002106  005200                          INC     R0              ;KEEP LSB SET
002110  103743                          BCS     2$              ;BR IF ZERO NOT TO CARRY YET
002112  012777  177001  177212          MOV     #177001,@DRVDBR ;TELL SLAVE TO GO TO NEXT TEST
002120  005077  177204                  CLR     @DRVCSR         ;TELL SLAVE TO READ TEST TERMINATOR
002124  032777  000400  177176  5$:     BIT     #400,@DRVCSR    ;IS IT THERE YET?
002132  001774                          BEQ     5$              ;WAIT FOR IT

                                .SBTTL  MTST2  TEST THAT SLAVE CAN ECHO THE 'STAT' BITS CORRECTLY
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;MASTER SENDS OUT A 'FNCT' CODE (1-7) TO THE SLAVE COMPUTER
                                ;THE MASTER THEN LOOKS FOR THE SLAVE TO ECHO THE CODE VIA THE
                                ;'STAT' BITS WITHIN A CERTAIN AMOUNT OF TIME
                                ;IF IT FAILS TO RETURN THE CORRECT CODE AN ERROR IS REPORTED
                                ;THIS TEST IS NOT EXITED UNTIL ALL 'FNCT' CODES HAVE BEEN
                                ;SENT AND RECEIVED CORRECTLY
                                ;;*****************************************************************
                                ;;*****************************************************************
002134                          MTST2:
002134  013737  001330  001122          MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
002142  012700  000002                  MOV     #2,R0           ;SET UP INITIAL FNCT BIT COUNT
002146  012737  001602  001124          MOV     #1602,$GDDAT    ;LD EXPECETD
002154  010077  177150          1$:     MOV     R0,@DRVCSR      ;LOAD FNCT BITS
```

```
                002160  032777  000400  177142  2$:     BIT     #400,@DRVCSR    ;LOOK FOR SLAVE TO ECHO VIA 'STAT' BITS
                002166  001774                          BEQ     2$              ;WAIT ON SLAVE
                002170  017737  177134  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
                002176  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
                002204  001402                          BEQ     3$              ;BR IF SO
                002206  104002                          ERROR   2               ;SLAVE FAILED TO ECHO 'STAT' BITS
                002210  000761                          BR      1$              ;LOOP ON ERROR ALWAYS
                002212  062737  001002  001124  3$:     ADD     #1002,$GDDAT    ;ADVANCE EXPECTED
                002220  062700  000002                  ADD     #2,R0           ;ADVANCE PTRN
                002224  032700  000020                  BIT     #20,R0          ;ALL BEEN DONE?
                002230  001751                          BEQ     1$              ;BR IF NOT
                002232  012777  177002  177072          MOV     #177002,@DRVDBR ;TELL SLAVE TO GO TO NEXT TEST
                002240  005077  177064                  CLR     @DRVCSR         ;SET THERE CYCLE
                002244  032777  000400  177056  4$:     BIT     #400,@DRVCSR    ;IS IT THERE YET?
                002252  001774                          BEQ     4$              ;WAIT ON SLAVE

                                        .SBTTL  MTST3  TEST THAT MASTER CAN XMIT A 32 WORD DATA BLOCK TO SLAVE
                                        ;;##################################################################
                                        ;;##################################################################
                                        ;MASTER XMITS A 32 WORD BLOCK OF DATA TO SLAVE
                                        ;THEN CHECKS FOR PROPER INTERRUPT STATUS, WC & BA
                                        ;THE SLAVE CHECKS THE SAME PLUS THE DATA RECEIVED
                                        ;THE TEST DOES NOT ADVANCE UNTIL A SUCCESSFUL XFER
                                        ;;##################################################################
                                        ;;##################################################################
                002254                  MTST3:
                002254  106427  000200  1$:     MTPS,   200             ;NO INTR ALLOWED YET
                002260  004537  003652          JSR     R5,SETVEC       ;SET UP INTR RETURN ADRS
                002264  002372                  3$                      ;RETURN TO 3$ ON INTR
                002266  004537  003730          JSR     R5,LDBUF        ;GO LOAD 'DBUF' WITH DATA PTRN
                002272  177740                  -32.                    ;DO 32. LOCATIONS
                002274  012737  100000  001340  MOV     #100000,TIME    ;SET UP A TIMER VALUE
                002302  013777  001360  177014  MOV     XPRAM,@DRVWCR   ;SET UP WORD COUNT
                002310  013777  001362  177010  MOV     XPRAM+2,@DRVBAR ;SET UP BUFFER ADRS
                002316  013777  001364  177004  MOV     XPRAM+4,@DRVCSR ;SET UP CONTROL - IE, FNCT 3 & GO(CLRS CYCLE)
                002324  106427  000000          MTPS,   0               ;ALLOW THE RDY INTR
                002330  005337  001340  2$:     DEC     TIME            ;WAIT FOR INTR
                002334  001375                  BNE     2$              ;UNTIL 0
                002336  017737  176766  001126  MOV     @DRVCSR,$BDDAT  ;READ CSR - SHOULD NEVER GET HERE
                002344  042777  000100  176756  BIC     #100,@DRVCSR    ;DISABLE IE
                002352  012737  005710  001124  MOV     #5710,$GDDAT    ;LD EXPECTED - STAT A & C, CYCLE, RDY, IE, FNCT 3
                002360  013737  001330  001122  MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                002366  104003                  ERROR   3               ;DATI FAILED TO CAUSE A WC INTR
                002370  000477                  BR      6$              ;GO RESYNC ON ERROR
                002372  022626          3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
                002374  012737  000040  001340  MOV     #40,TIME        ;SET UP WAIT DELAY
                002402  032777  000400  176720  11$:    BIT     #400,@DRVCSR    ;IS CYCLE SET?
                002410  001403                  BEQ     12$             ;YES
                002412  005337  001340          DEC     TIME            ;DECREMENT DELAY COUNT
                002416  001371                  BNE     11$
                002420  017737  176704  001126  12$:    MOV     @DRVCSR,$BDDAT  ;READ STATUS
                002426  042777  000100  176674  BIC     #100,@DRVCSR    ;DISABLE IE
                002434  012737  005710  001124  MOV     #5710,$GDDAT    ;LD EXPECTED - STAT A & C, CYCLE, RDY, IE, FNCT 3
                002442  023737  001124  001126  CMP     $GDDAT,$BDDAT   ;CORRECT?
```

```
002450  001405                             BEQ     4$                  ;BR IF SO
002452  013737  001330  001122             MOV     DRVCSR,$BDADR       ;SET UP CSR ADRS
002460  104004                             ERROR   4                   ;DATI STATUS ERROR
002462  000442                             BR      6$                  ;GO RESYNC ON ER
002464  005037  001124          4$:        CLR     $GDDAT              ;LD EXPECTED WC
002470  017737  176630  001126             MOV     @DRVWCR,$BDDAT      ;READ WORD COUNT
002476  001405                             BEQ     5$                  ;BR IF ZERO
002500  013737  001324  001122             MOV     DRVWCR,$BDADR       ;SET UP WCR ADRS
002506  104005                             ERROR   5                   ;DATI WORD COUNT ERROR
002510  000427                             BR      6$                  ;GO RESYNC ON ER
002512  013737  001352  001124  5$:        MOV     XPRAM+2,$GDDAT      ;GET STARTING ADRS OF XFER
002520  013700  001350                     MOV     XPRAM,R0            ;GET WC
002524  005400                             NEG     R0                  ;GET ACTUAL #
002526  006300                             ASL     R0                  ;CONVERT TO WORD
002530  060037  001124                     ADD     R0,$GDDAT           ;ADD TO BASE ADRS
002534  017737  176566  001126             MOV     @DRVBAR,$BDDAT      ;READ BAR ADRS
002542  042737  000001  001126             BIC     #BIT00,$BDDAT       ;ELIMINATE LSB
002550  023737  001124  001126             CMP     $GDDAT,$BDDAT       ;CORRECT?
002556  001416                             BEQ     8$                  ;BR IF SO
002560  013737  001326  001122             MOV     DRVBAR,$BDADR       ;SET UP BAR ADRS
002566  104006                             ERROR   6                   ;DATI BUFFER ADRS ERROR
002570  012737  000200  001340  6$:        MOV     #200,TIME           ;WAIT FOR SLAVE TO COMPLETE DATA CK
002576  005337  001340          7$:        DEC     TIME                ;COUNT AWAY
002602  001375                             BNE     7$                  ;UNTIL DONE
002604  112777  000002  176516             MOVB    #2,@DRVCSR          ;TELL SLAVE WE HAVE AN ERROR
002612  000720                             BR      1$                  ;REPEAT TEST ON ER
002614  012737  000100  001340  8$:        MOV     #100,TIME           ;WAIT FOR SLAVE TO COMPLETE CKS
002622  005337  001340          20$:       DEC     TIME                ;COUNT AWAY
002626  001375                             BNE     20$                 ;UNTIL DONE
002630  105077  176474                     CLRB    @DRVCSR             ;TELL SLAVE ALL OK
002634  017737  176470  001342  9$:        MOV     @DRVCSR,SAVE        ;READ CSR
002642  042737  170777  001342             BIC     #170777,SAVE        ;SAVE 'STAT' BITS
002650  001406                             BEQ     10$                 ;BR IF SLAVE DONE WITH NO ER'S
002652  022737  001000  001342             CMP     #1000,SAVE          ;WAS THERE AN ER?
002660  001365                             BNE     9$                  ;BR IF NOT
002662  000137  002254                     JMP     1$                  ;YES - REPEAT TEST ON ER
002666  004737  003672          10$:       JSR     PC,RSTVEC           ;GO RESTORE VECTOR

                                .SBTTL  STST1  RECEIVE MASTER'S DBR DATA AND SEND IT BACK VIA DBR
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;NOW THIS COMPUTER BECOMES THE SLAVE AND ECHOS MASTER'S DBR DATA
                                ;SLAVE COMPUTER STARTS HERE FROM PROGRAM START 204
                                ;;*****************************************************************
                                ;;*****************************************************************
002672                          STST1:
002672  005077  176432          1$:        CLR     @DRVCSR             ;TELL MASTER WE ARE READY
002676  032777  000400  176424  2$:        BIT     #400,@DRVCSR        ;DATA AVAILABLE?
002704  001774                             BEQ     2$                  ;WAIT ON IT
002706  017737  176420  001342             MOV     @DRVDBR,SAVE        ;GET DATA
002714  022737  177001  001342             CMP     #177001,SAVE        ;TEST TERMINATOR?
002722  001404                             BEQ     3$                  ;BR IF SO
002724  013777  001342  176400             MOV     SAVE,@DRVDBR        ;SEND IT BACK
002732  000757                             BR      1$                  ;GO LOOK FOR MORE DATA
```

```
                        3$:     NOP                     ;
                
                        .SBTTL  STST2  RECEIVE MASTER'S 'STAT' BITS AND SEND IT BACK VIA 'FNCT' BITS
                        ;;##################################################################
                        ;;##################################################################
                        ;RECEIVE 'STAT' BITS AND CONVERT TO 'FNCT' BITS AND WRITE TO CSR
                        ;;##################################################################
                        ;;##################################################################
                        STST2:
                1$:     BIC     #400,@DRVCSR    ;TELL MASTER WE ARE READY
                2$:     BIT     #400,@DRVCSR    ;LOOK FOR CYCLE
                        BEQ     2$              ;WAIT ON IT
                        CMP     #177002,@DRVDBR ;TEST TERMINATOR?
                        BEQ     3$              ;BR IF SO
                        MOV     @DRVCSR,SAVE    ;READ THE CSR
                        BIC     #170777,SAVE    ;SAVE ONLY THE STAT BITS
                        MOVB    SAVE+1,@DRVCSR  ;ECHO WITH FNCT BITS
                        BR      1$              ;LOOK FOR NEXT 'STAT' CODE
                3$:     CLR     @DRVCSR         ;TELL MASTER TO CONTINUE
                
                        .SBTTL  STST3  RECEIVE A 32 WORD BLOCK OF DATA AND MAKE STATUS & DATA CHECKS
                        ;;##################################################################
                        ;;##################################################################
                        ;THIS TEST SETS UP TO RECEIVE A 32 WORD BLOCK OF DATA FROM THE MASTER
                        ;THEN CHECKS FOR PROPER INTERRUPT STATUS, WC, BA & DATA
                        ;IF AN ERROR IS DETECTED THE SLAVE TELLS THE MASTER, REPORTS THE
                        ;ERROR, AND RESYNCS ON TEST
                        ;IF ALL OK THEN THIS COMPUTER BECOMES MASTER AND GOES TO MTST1
                        ;;##################################################################
                        ;;##################################################################
                        STST3:
                1$:     MTPS,   200             ;NO INTRS ALLOWED YET
                        JSR     R5,SETVEC       ;SET UP THE INTR RETURN ADRS
                        3$                      ;RETURN TO 3$ ON DATO INTR
                        JSR     R5,CLRBUF       ;GO CLR 'DBUF',
                        -32.                    ;DO 32. LOCATIONS
                        MOV     #100000,TIME    ;SET UP A TIMER VALUE
                        MOV     RPRAM,@DRVWCR   ;SET UP WORD COUNT
                        MOV     RPRAM+2,@DRVBAR ;SET UP BUFFER ADRS
                2$:     BIT     #400,@DRVCSR    ;LOOK FOR CYCLE
                        BEQ     2$              ;WAIT FOR IT
                        MOV     RPRAM+4,@DRVCSR ;SET UP CONTROL - IE, FNCT 3 & 1 & GO
                        MTPS,   0               ;ALLOW THE INTR
                4$:     DEC     TIME            ;WAIT FOR INTR
                        BNE     4$              ;UNTIL ZERO
                        MOV     @DRVCSR,$BDDAT  ;READ CSR - SHOULD NEVER GET HERE
                        BIC     #100,@DRVCSR    ;DISABLE IE
                        MOV     #4312,$BDDAT    ;LD EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
                        MOV     DRVCSR,$BDADR   ;SET UP CSR ADRS
                        ERROR   7               ;DATO FAILED TO CAUSE A WC INTR
                        BR      10$             ;GO RESYNC ON ER
                3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RETURN
                        MOV     @DRVCSR,$BDDAT  ;READ STATUS
                        BIC     #100,@DRVCSR    ;DISABLE IE
```

```
003160  012737  004312  001124          MOV     #4312,$GDDAT        ;LD EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
003166  023737  001124  001126          CMP     $GDDAT,$BDDAT       ;CORRECT?
003174  001405                          BEQ     4$                  ;BR IF SO
003176  013737  001330  001122          MOV     DRVCSR,$BDADR       ;SET UP CSR ADRS
003204  104010                          ERROR   10                  ;DATO STATUS ERROR
003206  000474                          BR      10$                 ;GO RESYNC ON ERROR
003210  005037  001124          4$:     CLR     $GDDAT              ;LD EXPECTED WC
003214  017737  176104  001126          MOV     @DRVWCR,$BDDAT      ;READ WCR
003222  001405                          BEQ     5$                  ;BR IF SO
003224  013737  001324  001122          MOV     DRVWCR,$BDADR       ;SET UP WCR ADRS
003232  104011                          ERROR   11                  ;DATO WORD COUNT ERROR
003234  000461                          BR      10$                 ;GO RESYNC ON ERROR
003236  013737  001360  001124  5$:     MOV     RPRAM+2,$GDDAT      ;GET STARTING ADRS OF XFER
003244  013700  001356                  MOV     RPRAM,R0            ;GET WC
003250  005400                          NEG     R0                  ;GET ACTUAL #
003252  006300                          ASL     R0                  ;CONVERT TO WORD
003254  060037  001124                  ADD     R0,$GDDAT           ;ADD TO BASE ADRS
003260  017737  176042  001126          MOV     @DRVBAR,$BDDAT      ;READ BAR ADRS
003266  042737  000001  001126          BIC     #BIT00,$BDDAT       ;ELIMINATE LSB
003274  023737  001124  001126          CMP     $GDDAT,$BDDAT       ;CORRECT?
003302  001405                          BEQ     6$                  ;BR IF SO
003304  013737  001326  001122          MOV     DRVBAR,$BDADR       ;SET UP BAR ADRS
003312  104012                          ERROR   12                  ;DATO BUFFER ADRS ERROR
003314  000431                          BR      10$                 ;GO RESYNC ON ERROR
003316  013700  001360          6$:     MOV     RPRAM+2,R0          ;GET BUFFER ADRS
003322  013701  001356                  MOV     RPRAM,R1            ;GET WORD COUNT
003326  012702  177776          7$:     MOV     #177776,R2          ;GET 1ST DATA PTRN (FLOATING 0)
003332  005003                          CLR     R3                  ;R3 SAYS WHEN TO SHIFT PTRN
003334  020220                  8$:     CMP     R2,(R0)+            ;COMPARE DATA IN DBUF TO EXPECTED
003336  001011                          BNE     9$                  ;BR IF DATA ERROR
003340  005201                          INC     R1                  ;COUNT THE WORD COUNT
003342  001431                          BEQ     13$                 ;BR IF DATA CHECKS DONE
003344  005102                          COM     R2                  ;CONVERT PTRN TO FLOATING 1
003346  005103                          COM     R3                  ;TIME TO SHIFT?
003350  001371                          BNE     8$                  ;BR IF NOT - GO CK NEXT
003352  006302                          ASL     R2                  ;FLOAT PTRN LEFT
003354  005202                          INC     R2                  ;KEEP LSB SET
003356  103363                          BCC     7$                  ;GO RESET FLOATING PTRN
003360  000765                          BR      8$                  ;GO CHECK NEXT
003362  014037  001126          9$:     MOV     -(R0),$BDDAT        ;GET BAD DATA
003366  010037  001122                  MOV     R0,$BDADR           ;GET MEM ADRS OF DATA ER
003372  010237  001124                  MOV     R2,$GDDAT           ;LD EXPECTED DATA
003376  104013                          ERROR   13                  ;DATO DATA ERROR
003400  012737  000100  001340  10$:    MOV     #100,TIME           ;LET MASTER FINISH ITS CHECKS
003406  005337  001340          11$:    DEC     TIME                ;COUNT AWAY
003412  001375                  12$:    BNE     11$                 ;UNTIL DONE
003414  112777  000002  175706          MOVB    #2,@DRVCSR          ;TELL MASTER WE HAVE AN ERROR
003422  000137  003014                  JMP     1$                  ;DO TEST AGAIN ON ERROR
003426  105077  175676          13$:    CLRB    @DRVCSR             ;TELL MASTER ALL OK
003432  017737  175672  001342  14$:    MOV     @DRVCSR,SAVE        ;READ CSR
003440  042737  170777  001342          BIC     #170777,SAVE        ;SAVE 'STAT' BITS ONLY
003446  001406                          BEQ     EOPT                ;BR IF MASTER DONE WITH NO ER'S
003450  022737  001000  001342          CMP     #1000,SAVE          ;DOES MASTER HAVE AN ERROR?
003456  001365                          BNE     14$                 ;BR IF NOT
```

```
                                                JMP     1$              ;YES - DO TEST AGAIN ON ERROR
003460  000137  003014                  EOPT:   JSR     PC,RSTVEC       ;GO RESTORE VECTOR
003464  004737  003672                          CKSWR                   ;GO CHECK SWR
003470  104406                                  TST     MSTER           ;WERE WE STARTED AS MASTER?
003472  005737  001344                          BNE     EOPTA           ;BR IF NOT
003476  001055                                  DEC     ICOUNT          ;COUNT TEST PASS
003500  005337  001346                          BNE     EOPTA           ;BR IF NOT DUE FOR 'END PASS' MSG
003504  001052                                  MOV     #3000,ICOUNT    ;RESET PASS COUNT
003506  012737  003000  001346          .SBTTL  END OF PASS ROUTINE

                                        ;;**************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO EOPTA

003514                                  SEOP:
003514  000240                                  NOP
003516  005037  001102                          CLR     $ISTNM          ;;ZERO THE TEST NUMBER
003522  005237  001100                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
003526  042737  100000  001100                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
003534  005327                                  DEC     (PC)+           ;;LOOP?
003536  000001                  SEOPCT: .WORD   1
003540  003022                          BGT     $DOAGN                  ;;YES
003542  012737                          MOV     (PC)+,@(PC)+            ;;RESTORE COUNTER
003544  000001                  SENDCT: .WORD   1
003546  003536                          $EOPCT
003550  104400  003615                  TYPE    ,SENDMG                 ;;TYPE "END PASS #"
003554  013746  001100                  MOV     $PASS,-(SP)             ;;SAVE $PASS FOR TYPEOUT
003560  104404                          TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
003562  104400  003612                  TYPE    ,$ENULL                 ;;TYPE A NULL CHARACTER
003566  013700  000042          SGET42: MOV     @#42,R0                 ;;GET MONITOR ADDRESS
003572  001405                          BEQ     $DOAGN                  ;;BRANCH IF NO MONITOR
003574  000005                          RESET                           ;;CLEAR THE WORLD
003576  004710                  SENDAD: JSR     PC,(R0)                 ;;GO TO MONITOR
003600  000240                          NOP                             ;;SAVE ROOM
003602  000240                          NOP                             ;;FOR
003604  000240                          NOP                             ;;ACT11
003606                          SDOAGN:
003606  000137                          JMP     @(PC)+                  ;;RETURN
003610  003632                  $RTNAD: .WORD   EOPTA
003612      377         377     $ENULL: .BYTE   -1,-1,0                 ;;NULL CHARACTER STRING
003615      015  042412  042116 SENDMG: .ASCIZ  <15><12>/END PASS #/
003622  050040  051501  020123
003630  000043
003632  012737  001000  001340  EOPTA:  MOV     #1000,TIME              ;SET UP A COUNT
003640  005337  001340  EOPTB:  DEC     TIME                    ;STALL FOR OTHER CPU TO BECOME SLAVE
003644  001375                          BNE     EOPTB                   ;UNTIL TIME=0
003646  000137  002012                  JMP     MTST1                   ;NOW BECOME MASTER
```

```
                                .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
                                ;;*****************************************************************
003652  106427  000200          SETVEC: MTPS,   200             ;SET UP FOR NO INTERRUPT
003656  012577  175452                  MOV     (R5)+,@DRVCT0   ;SET UP INTR RETURN ADRS
003662  012777  000200  175446          MOV     #200,@DRVCT2    ;KEEP PRIORITY LEVEL AT TOP ON INTR
003670  000205                          RTS     R5              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE RESTORES THE DRV11B INTERRUPT VECTOR TO A HALT
                                ;AND RAISES THE PRIORITY LEVEL
                                ;;*****************************************************************
003672  013777  001336  175434  RSTVEC: MOV     DRVCT2,@DRVCT0  ;POINT VECTOR TO HALT
003700  005077  175432                  CLR     @DRVCT2         ;SET UP HALT
003704  106427  000200                  MTPS,   200             ;RAISE PRIORITY LEVEL
003710  000207                          RTS     PC              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE CLEARS THE 'DBUF' BEFORE A 'DATI' XFER
                                ;THE # OF LOCATIONS IN 'DBUF' TO BE CLEARED IS SPECIFIED BY
                                ;THE VALUE IN THE CALL +2 - WHEN ALL CLEARED THE RETURN IS TO
                                ;THE CALL +4
                                ;;*****************************************************************
003712  012500                  CLRBUF: MOV     (R5)+,R0        ;GET THE LOC COUNT
003714  012701  006610                  MOV     #DBUF,R1        ;GET 1ST ADRS
003720  005021                  1$:     CLR     (R1)+           ;CLR MEM LOC
003722  005200                          INC     R0              ;COUNT LOC
003724  001375                          BNE     1$              ;UNTIL ALL DONE
003726  000205                          RTS     R5              ;EXIT

                                ;;*****************************************************************
                                ;THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
                                ;THE # OF LOCATIONS IN 'DBUF' TO BE LOADED IS SPECIFIED BY
                                ;THE VALUE IN THE CALL +2 - WHEN ALL LOADED THE RETURN IS TO CALL +4
                                ;;*****************************************************************
003730  012500                  LDBUF:  MOV     (R5)+,R0        ;GET LOC COUNT
003732  012701  006610                  MOV     #DBUF,R1        ;GET 1ST ADRS
003736  012702  177776          1$:     MOV     #177776,R2      ;SET UP FLOATING ZERO PTRN
003742  005003                          CLR     R3              ;R3 SAYS WHEN TO SHIFT PTRN
003744  010221                  2$:     MOV     R2,(R1)+        ;LOAD MEM WITH PTRN
003746  005200                          INC     R0              ;COUNT LOC
003750  001001                          BNE     3$              ;BR IF MORE
003752  000205                          RTS     R5              ;EXIT
003754  005102                  3$:     COM     R2              ;CONVERT PTRN TO FLOATING 1
003756  005103                          COM     R3              ;SHOULD WE SHIFT?
003760  001371                          BNE     2$              ;BR IF NOT - THIS WILL BE A FLOATING 1
003762  006302                          ASL     R2              ;FLOAT ZERO LEFT
003764  005202                          INC     R2              ;KEEP LSB SET
003766  103363                          BCC     1$              ;GO RESET FLOATING PATRN
003770  000765                          BR      2$              ;GO LOAD NEXT PATRN
```

```
                                        .SBTTL  SYSMAC ROUTINES

                                .SBTTL  TYPE ROUTINE

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

003772  105737  001157          STYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
003776  100002                          BPL     1$              ;;BR IF YES
004000  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
004002  000407                          BR      3$              ;;LEAVE
004004  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
004006  017600  000002                  MOV     2(SP),R0        ;;GET ADDRESS OF ASCIZ STRING
004012  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
004014  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
004016  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
004020  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
004022  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
004026  000002                          RTI                     ;;RETURN
004030  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
004034  001430                          BEQ     8$
004036  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
004042  001006                          BNE     5$
004044  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
004046  104400                          TYPE                    ;;TYPE A CR AND LF
004050  001165                          $CRLF
004052  105037  004206                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
004056  000755                          BR      2$              ;;GET NEXT CHARACTER
004060  004737  004142          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
004064  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
004070  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
004072  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
004076  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
004102  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
004104  004737  004142                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
004110  105337  004206                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
004114  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

004116  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
1115  004122  004737  004142          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1116  004126  132737  000007  004206          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1117  004134  001372                          BNE     9$              ;;TAB STOP
1118  004136  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
1119  004140  000724                          BR      2$              ;;GET NEXT CHARACTER
1120  004142  105777  175002          $TYPEC: TSTB    @#$TPS          ;;WAIT UNTIL PRINTER IS READY
1121  004146  100375                          BPL     $TYPEC
1122  004150  116677  000002  174774          MOVB    2(SP),@#$TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1123  004156  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1124  004164  001003                          BNE     1$              ;;BRANCH IF NO
1125  004166  105037  004206                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1126  004172  000406                          BR      $TYPEX          ;;EXIT
1127  004174  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1128  004202  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
1129  004204  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
1130  004206  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
1131  004210  000207          $TYPEX: RTS     PC
1132
1133                          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1134
1135                          ;;#################################################################
1136                          ;#THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1137                          ;#OCTAL (ASCII) NUMBER AND TYPE IT.
1138                          ;#$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1139                          ;#CALL:
1140                          ;#      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1141                          ;#      TYPOS                   ;;CALL FOR TYPEOUT
1142                          ;#      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1143                          ;#      .BYTE   M               ;;M=1 OR 0
1144                          ;#                                      ;;1=TYPE LEADING ZEROS
1145                          ;#                                      ;;0=SUPPRESS LEADING ZEROS
1146                          ;#
1147                          ;#$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1148                          ;#$TYPOS OR $TYPOC
1149                          ;#CALL:
1150                          ;#      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1151                          ;#      TYPON                   ;;CALL FOR TYPEOUT
1152                          ;#
1153                          ;#$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1154                          ;#CALL:
1155                          ;#      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1156                          ;#      TYPOC                   ;;CALL FOR TYPEOUT
1157
1158  004212  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
1159  004216  116637  000001  004435          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
1160  004224  116637  004437                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
1161  004230  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
1162  004234  000406                          BR      $TYPON
1163  004236  112737  000001  004435  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
1164  004244  112737  000006  004437          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
1165  004252  112737  000005  004434  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
1166  004260  010346                          MOV     R3,-(SP)        ;;SAVE R3
1167  004262  010446                          MOV     R4,-(SP)        ;;SAVE R4
1168  004264  010546                          MOV     R5,-(SP)        ;;SAVE R5
```

```
                113704  004437          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                005404                  NEG     R4
                062704  000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
                110467  004436          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
                113704  004435          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
                016505  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
                005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
                006105          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
                000404                  BR      3$              ;;GO DO MSB
                006105          2$:     ROL     R5              ;;FORM THIS DIGIT
                006105                  ROL     R5
                006105                  ROL     R5
                010503                  MOV     R5,R3
                006103          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
                105367  004436          DECB    $OMODE          ;;TYPE THIS DIGIT?
                100016                  BPL     7$              ;;BR IF NO
                042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
                001002                  BNE     4$              ;;TEST FOR 0
                005704                  TST     R4              ;;SUPPRESS THIS 0?
                001403                  BEQ     5$              ;;BR IF YES
                005204          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
                052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
                052703  000040  5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
                110367  004432          MOVB    R3,6$           ;;SAVE FOR TYPING
                104400  004432          TYPE    ,6$             ;;GO TYPE THIS DIGIT
                105367  004434  7$:     DECB    $OCNT           ;;COUNT BY 1
                003347                  BGT     2$              ;;BR IF MORE TO DO
                005702                  TST     R2              ;;BR IF DONE
                005204                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
                000744                  BR      2$              ;;GO DO THE LAST DIGIT
                012605          6$:     MOV     (SP)+,R5        ;;RESTORE R5
                012604                  MOV     (SP)+,R4        ;;RESTORE R4
                012603                  MOV     (SP)+,R3        ;;RESTORE R3
                011646  000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                012616                  MOV     (SP)+,(SP)
                000002                  RTI                     ;;RETURN
                000000          6$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                000000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                000000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                000000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
                000000          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                
                ;;#################################################################
                ;#THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                ;#SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                ;#NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                ;#BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                ;#REPLACED WITH SPACES.
                ;#CALL:
                ;#      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                ;#      TYPOS                   ;;GO TO THE ROUTINE
                
004440                  STYPOS:
```

```
                                                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
                                                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
                                020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
                                000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
                                                        BPL     1$              ;;BR IF INPUT IS POS.
                                                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
                                000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
                                                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
                                004654                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
                                000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
                                                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
                                004644                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
                                                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
                                                        BLT     4$              ;;BR IF DONE
                                                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
                                                        BR      3$
                                                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
                                                        TST     R2              ;;CHECK IF BCD DIGIT=0
                                                        BNE     5$              ;;FALL THROUGH IF 0
                                                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
                                                        BMI     7$              ;;BR IF YES
                                                5$:     ASLB    (SP)            ;;MSD?
                                                        BCC     6$              ;;BR IF NO
                                000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
                                000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
                                000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
                                                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
                                                        TST     (R0)+           ;;JUST INCREMENTING
                                000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
                                                        BLT     2$              ;;GO DO THE NEXT DIGIT
                                                        BGT     8$              ;;GO TO EXIT
                                                        MOV     R5,R2           ;;GET THE LSD
                                                        BR      6$              ;;GO CHANGE TO ASCII
                                                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
                                                        BPL     9$              ;;BR IF NO
                                177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
                                                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
                                                        MOV     (SP)+,R5        ;;POP STACK INTO R5
                                                        MOV     (SP)+,R3        ;;POP STACK INTO R3
                                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                                004654                  TYPE    $DBLK           ;;NOW TYPE THE NUMBER
                                000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
                                                        MOV     (SP)+,(SP)
                                                        RTI                     ;;RETURN TO USER

                                                SDTBL:  10000.
                                                        1000.
                                                        100.
                                                        10.
004654                          000004          SDBLK:  .BLKW   4
```

```
                                        .SBTTL  ERROR HANDLER ROUTINE

                                        ;;**********************************************************************
                                        ;#THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;#SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;#AND GO TO $WRCK ON ERROR
                                        ;#THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;#SW15=1         HALT ON ERROR
                                        ;#SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;#SW10=1         BELL ON ERROR
                                        ;#CALL
                                        ;#      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

                                        $ERROR:
                                                CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
                                                CKSWR                   ;GO LOOK FOR SWR CHANGE
                                        7$:     INCB    $ERRFLG         ;;SET THE ERROR FLAG
                                                BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
                                                MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
                                                BIT     #BIT10,@SWR     ;;BELL ON ERROR?
                                                BEQ     1$              ;;NO - SKIP
                                                TYPE    ,$BELL          ;;RING BELL
                                        1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
                                                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
                                                SUB     #2,$ERRPC
                                                MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
                                                BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
                                                BNE     20$             ;;SKIP TYPEOUTS
                                                JSR     PC,$WRCK        ;;GO TO USER ERROR ROUTINE
                                                TYPE    ,$CRLF
                                        20$:
                                        2$:     TST     @SWR            ;;HALT ON ERROR
                                                BPL     3$              ;;SKIP IF CONTINUE
                                                HALT                    ;;HALT ON ERROR!
                                                CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
                                        3$:
                                                RTI                     ;;RETURN
                                        ;;**********************************************************************
                                        ;GO TYPE ERROR
                                        ;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
                                        ;;**********************************************************************
005000  004737  005010                  $WRCK:  JSR     PC,$ERRTYP      ;GO TYPE ERROR
005004  104406                                  CKSWR                   ;GO LOOK FOR SWR CHANGE
005006  000207                                  RTS     PC              ;RETURN TO ERROR HANDLER
                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                        ;;**********************************************************************
                                        ;#THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;#ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;#AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

005010                                  $ERRTYP:
005010  104400  001165                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
005014  010046                                  MOV     R0,-(SP)        ;;SAVE R0
```

```
1331  005016  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
1332  005020  153700  001114                  BISB    @#SITEMB,R0
1333  005024  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
1334                                                                  ;;TYPE THE PC OF THE ERROR
1335  005026  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
1336                                                                  ;;ERROR ADDRESS
1337  005032  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1340  005034  000426                          BR      6$              ;;GET OUT
1341  005036  006300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
1342  005040  006300                          ASL     R0              ;;       WORK FOR THE ERROR TABLE
1343  005042  006300                          ASL     R0
1344  005044  006300                          ASL     R0
1345  005046  062700  001170                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
1346  005052  012037  005062                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
1347  005056  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
1350  005060  104400                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
1351  005062  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
1352  005064  104400  001165                  TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
1353  005070  012037  005100          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
1354  005074  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
1355  005076  104400                          TYPE                    ;;TYPE THE "DATA HEADER"
1356  005100  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
1357  005102  104400  001165                  TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
1360  005106  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
1361  005110  001004                          BNE     7$              ;;GO TYPE THE DATA
1362  005112  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
1363  005114  104400  001165                  TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
1364  005120  000207                          RTS     PC              ;;RETURN
1365                                  7$:
1366  005122  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
1367  005124  104401                          TYPOC                   ;;GO TYPE--OCTAL ASC"I(ALL DIGITS)
1370  005126  005710                          TST     (R0)            ;;IS THERE ANOTHER №    ??
1371  005130  001770                          BEQ     6$              ;;BR IF NO
1372  005132  104400  005140                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
1373  005136  000771                          BR      7$              ;;LOOP
1374  005140  020040  000           8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
1375  005144                                  .EVEN
1376                                  .SBTTL  SCOPE HANDLER ROUTINE
1377
1400                                  ;;**************************************************************
1401                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS, IT WILL INCREMENT
1402                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1403                                  ;*AND LOAD THE ERROR FLAG (SERFLG) INTO DISPLAY<15:08>
1404                                  ;*CALL
1405                                  ;*      SCOPE           ;;SCOPE=IOT
1406
1407  005144                          $SCOPE:
1410  005144  104406                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
1411  005146  104406                          CKSWR                   ;GO LOOK FOR SWR CHANGE
1412                                  ;#####START OF CODE FOR THE XOR TESTER#####
1413  005150  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1414                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1415  005152  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1416  005156  012737  005176  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
```

```
1385  005164  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
1386  005170  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1387  005174  000404                            BR      $SVLAD          ;;GO TO THE NEXT TEST
1388  005176  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
1389  005200  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1390  005204  000406                            BR      $OVER           ;;LOOP ON THE PRESENT TEST
1391  005206                            6$:;####END OF CODE FOR THE XOR TESTER#####
1392  005206  105237  001102            $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
1393  005212  011637  001106                    MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
1394  005216  105037  001103                    CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
1395  005222  013777  001102  173712    $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
1396  005230  013716  001106                    MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
1397  005234  000002                            RTI                     ;;FIXES PS
1398                                    .SBTTL  TTY INPUT ROUTINE
1399
1400                                    ;;**************************************************************
1401                                    .ENABL  LSB
1402
1403                                    ;;**************************************************************
1404                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1405                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1406                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1407                                    ;*WHEN OPERATING IN TTY FLAG MODE.
1408  005236  022737  000176  001140    $CKSWR: CMP     @#SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
1409  005244  001074                            BNE     15$             ;;BRANCH IF NO
1410  005246  105777  173672                    TSTB    @#TKS           ;;CHAR THERE?
1411  005252  100071                            BPL     15$             ;;IF NO, DON'T WAIT AROUND
1412  005254  117746  173666                    MOVB    @#TKB,-(SP)     ;;SAVE THE CHAR
1413  005260  042716  177600                    BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
1414  005264  022726  000007                    CMP     #7,(SP)+        ;;IS IT A CONTROL G?
1415  005270  001062                            BNE     15$             ;;NO, RETURN TO USER
1416  005272  123727  001134  000001            CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
1417  005300  001456                            BEQ     15$             ;;BRANCH IF YES
1418
1419  005302  104400  005763                    TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (↑G)
1420  005306  104400  005770            $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
1421  005312  013746  000176                    MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
1422  005316  104401                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1423  005320  104400  006001                    TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
1424  005324  005046                    19$:    CLR     -(SP)           ;;CLEAR COUNTER
1425  005326  005046                            CLR     -(SP)           ;;THE NEW SWR
1426  005330  105777  173610            7$:     TSTB    @#TKS           ;;CHAR THERE?
1427  005334  100375                            BPL     7$              ;;IF NOT TRY AGAIN
1428
1429  005336  117746  173604                    MOVB    @#TKB,-(SP)     ;;PICK UP CHAR
1430  005342  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1431
1432
1433
1434  005346  021627  000025            9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1435  005352  001005                            BNE     10$             ;;BRANCH IF NOT
1436  005354  104400  005756                    TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
1437  005360  062706  000006            20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1438  005364  000757                            BR      19$             ;;LET'S TRY IT AGAIN
```

```
1439
1440
1441  005366  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
1442  005372  001022                          BNE     16$             ;;BRANCH IF NO
1443  005374  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
1444  005400  001403                          BEQ     11$             ;;BRANCH IF YES
1445  005402  016677  000002  173530          MOV     2(SP),@SWR      ;;SAVE NEW SWR
1446  005410  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
1447  005414  104400  001165          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
1448  005420  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
1449  005426  001003                          BNE     15$             ;;BRANCH IF NOT
1450  005430  012777  000100  173506          MOV     #100,@STKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1451  005436  000002          15$:    RTI                             ;;RETURN
1452  005440  004737  004142          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1453  005444  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
1454  005450  002420                          BLT     18$             ;;BRANCH IF YES
1455  005452  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
1456  005456  003015                          BGT     18$             ;;BRANCH IF YES
1457  005460  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1458  005464  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
1459  005470  001403                          BEQ     17$             ;;BRANCH IF YES
1460  005472  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
1461  005474  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
1462  005476  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
1463  005500  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1464  005504  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1465  005510  000707                          BR      7$              ;;GET THE NEXT ONE
1466  005512  104400  001164          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
1467  005516  000720                          BR      20$             ;;SIMULATE CONTROL-U
1468                                  .DSABL  LSB
1469
1470
1471                                  ;;**********************************************************************
1472                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1473                                  ;*CALL:
1474                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
1475                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1476                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
1477                                  ;
1478
1479  005520  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
1480  005522  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
1481  005530  105777  173410          1$:     TSTB    @#TKS           ;;WAIT FOR
1482  005534  100375                          BPL     1$              ;;A CHARACTER
1483  005536  117766  173404  000004          MOVB    @#TKB,4(SP)     ;;READ THE TTY
1484  005544  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
1485  005552  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
1486  005560  001013                          BNE     3$              ;;BRANCH IF NO
1487  005562  105777  173356          2$:     TSTB    @#TKS           ;;WAIT FOR A CHARACTER
1488  005566  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
1489  005570  117746  173352                  MOVB    @#TKB,-(SP)     ;;GET CHARACTER
1490  005574  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1491  005600  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
1492  005604  001366                          BNE     2$              ;;IF NOT DISCARD IT
```

```
1493  005606  000750                        BR      1$                      ;;YES, RESUME
1494  005610  026627  000004  000140  3$:   CMP     4(SP),#140              ;;IS IT UPPER CASE?
1495  005616  002407                        BLT     4$                      ;;BRANCH IF YES
1496  005620  026627  000004  000175        CMP     4(SP),#175              ;;IS IT A SPECIAL CHAR?
1497  005626  003003                        BGT     4$                      ;;BRANCH IF YES
1498  005630  042766  000040  000004        BIC     #40,4(SP)               ;;MAKE IT UPPER CASE
1499  005636  000002                4$:     RTI                             ;;GO BACK TO USER
1500                                        ;;**************************************************************
1501                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1502                                        ;*CALL:
1503                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1504                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1505                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1506
1507  005640  010346                $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
1508  005642  012703  005746        1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
1509  005646  022703  005756        2$:     CMP     #$TTYIN+8.,R3           ;;BUFFER FULL?
1510  005652  101405                        BLOS    4$                      ;;BR IF YES
1511  005654  104407                        RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
1512  005656  112613                        MOVB    (SP)+,(R3)              ;;GET CHARACTER
1513  005660  122713  000177        10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
1514  005664  001003                        BNE     3$                      ;;SKIP IF NOT
1515  005666  104400  001164        4$:     TYPE    $QUES                   ;;TYPE A '?'
1516  005672  000763                        BR      1$                      ;;CLEAR THE BUFFER AND LOOP
1517  005674  111337  005744        3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
1518  005700  104400  005744                TYPE    9$
1519  005704  122723  000015                CMPB    #15,(R3)+               ;;CHECK FOR RETURN
1520  005710  001365                        BNE     2$                      ;;LOOP IF NOT RETURN
1521  005712  105063  177777                CLRB    -1(R3)                  ;;CLEAR RETURN (THE 15)
1522  005716  104400  001166                TYPE    $LF                     ;;TYPE A LINE FEED
1523  005722  012603                        MOV     (SP)+,R3                ;;RESTORE R3
1524  005724  011646                        MOV     (SP),-(SP)              ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1525  005726  016666  000004  000002        MOV     4(SP),2(SP)             ;;      FIRST ASCII CHARACTER ON IT
1526  005734  012766  005746  000004        MOV     #$TTYIN,4(SP)
1527  005742  000002                        RTI                             ;;RETURN
1528  005744  000                   9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
1529  005745  000                           .BYTE   0                       ;;TERMINATOR
1530  005746  000010                $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
1531  005756  052536  005015  000   $CNTLU: .ASCIZ  /↑U/<15><12>            ;;CONTROL "U"
1532  005763  136  006507  000012   $CNTLG: .ASCIZ  /↑G/<15><12>            ;;CONTROL "G"
1533  005770  005015  053523  020122  $WHAR:  .ASCIZ  <15><12>/WAR = /
1534  005776  020075  000
1535  006001  040  047040  053E05     $WNEW:  .ASCIZ  /  NEW = /
1536  006006  036440  000040
1537                                   .EVEN
1538                                   .SBTTL  TRAP DECODER
1539
1540                                   ;;**************************************************************
1541                                   ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1542                                   ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1543                                   ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1544                                   ;*GO TO THAT ROUTINE.
1545
1546  006012  010046                $TRAP:  MOV     R0,-(SP)                ;;SAVE R0
```

```
                                        MOV     2(SP),R0        ;;GET TRAP ADDRESS
                                        TST     -(R0)           ;;BACKUP BY 2
                                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
                                        ASL     R0              ;;POSITION FOR INDEXING
006026  016000  006034                  MOV     STRPAD(R0),R0   ;;INDEX TO TABLE
006032  000200                          RTS     R0              ;;GO TO ROUTINE

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
006034                          $TRPAD:
006034  003772                          $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
006036  004236                          $TYPOC  ;;CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
006040  004212                          $TYPOS  ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
006042  004252                          $TYPON  ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
006044  004440                          $TYPDS  ;;CALL=TYPDS    TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)

006046  005306                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

006050  005236                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
006052  005520                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
006054  005640                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE

                                .SBTTL  ASCII MESSAGES
006056  046123  053101  020105  EM1:    .ASCIZ  /SLAVE FAILED TO ECHO DBR CONTENTS/
006064  040523  046111  042105
006072  052040  020117  041505
006100  047510  042040  051102
006106  041440  047117  042524
006114  052116  000123
006120  046123  053101  020105  EM2:    .ASCIZ  /SLAVE FAILED TO ECHO 'STAT' BITS/
006126  040523  046111  042105
006134  052040  020117  041505
006142  047510  023440  052123
006150  052101  020047  044502
006156  051524      000
006161      105 044501  042514  EM3:    .ASCIZ  /FAILED TO INTR ON A 'DATI'/
006166  020104  047524  044440
006174  052116  020122  047117
006202  040440  023440  040504
006210  044524  000047
006214  052123  052101  051525  EM4:    .ASCIZ  /STATUS ER ON 'DATI'/
006222  040440  020122  047117
006230  023440  040504  044524
006236  000047
006240  046527  042122  041440  EM5:    .ASCIZ  /WORD COUNT ER ON 'DATI'/
006246  052517  052116  042440
006254  020122  047117  023440
006262  040504  044524  000047
006270  052502  043106  051105  EM6:    .ASCIZ  /BUFFER ADRS ER ON 'DATI'/
```

```
 1601  006276  040440  051104  020123
 1602  006304  051105  047440  020116
 1603  006312  042047  052101  023511
 1604  006320      000
 1605  006321      105  044501  042514  EM7:    .ASCIZ  /FAILED TO INTR ON A 'DATO'/
 1606  006336  020104  047524  040440
 1607  006354  052116  020122  047117
 1608  006372  040440  023440  040504
 1609  006410  047524  000047
 1610  006424  052123  052101  051525  EM10:   .ASCIZ  /STATUS ER ON 'DATO'/
 1611  006442  040440  020122  047117
 1612  006460  023440  040504  047524
 1613  006476  000047
 1614  006500  047527  042122  041440  EM11:   .ASCIZ  /WORD COUNT ER ON 'DATO'/
 1615  006516  052517  052116  042440
 1616  006534  020122  047117  023440
 1617  006552  040504  047524  000047
 1618  006570  052502  043106  051105  EM12:   .ASCIZ  /BUFFER ADRS ER ON 'DATO'/
 1619  006606  040440  051104  020123
 1620  006624  051105  047440  020116
 1621  006642  042047  052101  023517
 1622  006660      000
 1623  006661      104  052101  020101  EM13:   .ASCIZ  /DATA ER ON 'DATO'/
 1624  006676  051105  047440  020116
 1625  006714  042047  052101  023517
 1626  006732      000
 1627  006733      105  051122  041520  DH1:    .ASCIZ  /ERRPC   BUSADR  EXPCT   RCVD/
 1628  006751  020040  041040  051523
 1629  006767  042101  020122  042440
 1630  007005  050130  052103  020040
 1631  007023  051040  053103  000104
 1632  007041  051105  020122  020103  DH2:    .ASCIZ  /ERRPC   MEMADR  EXPCT   RCVD/
 1633  007057  020040  046515  040515
 1634  007075  051104  020040  054105
 1635  007113  041520  020124  020040
 1636  007131  041522  042126      000
 1637
 1638  007136                   .EVEN
 1639  007136  001116  001122  001124  DT1:    $ERRPC,$BDADR,$GDDAT,$BDDAT,0
 1640  007144  001126  000000
 1641                   ;;#################################################################
 1642                   ;'DBUF' IS THE WORKING AREA IN MEM FOR ALL NPR OPERATIONS
 1643                   ;;#################################################################
 1644  007150  000000  DBUF:   0               ;1ST ADRS OF DATA BUFFER
 1645  007152  000001          .END
```

DVDRBA.P11 CROSS REFERENCE TABLE

```
BIT0    = 000001          431#
BIT00   = 000001          421#    431     812     922
BIT01   = 000002          420#    430
BIT02   = 000004          419#    429
BIT03   = 000010          418#    428
BIT04   = 000020          417#    427
BIT05   = 000040          416#    426
BIT06   = 000100          415#    425
BIT07   = 000200          414#    424
BIT08   = 000400          413#    423
BIT09   = 001000          412#    422
BIT1    = 000002          430#
BIT10   = 002000          411#   1296
BIT11   = 004000          410#
BIT12   = 010000          409#
BIT13   = 020000          408#   1303
BIT14   = 040000          407#
BIT15   = 100000          406#
BIT2    = 000004          429#
BIT3    = 000010          428#
BIT4    = 000020          427#
BIT5    = 000040          426#
BIT6    = 000100          425#
BIT7    = 000200          424#
BIT8    = 000400          423#
BIT9    = 001000          422#
BPTVEC= 000014            438#
CKSWR   = 104406          960    1291    1292    1311    1319    1378    1379    1570#
CLRBUF  = 043712          884    1034#
CR      = 000015          346#   1123    1133
CRLF    = 000200          347#    679    1094    1133
DBUF    = 006610          611     614    1035    1047    1644#
DDISP   = 177570          353#    490     640
DH1     = 006503          521     527     533     539     544     550     556     562     568     574    1627#
DH2     = 006540          580    1632#
DISPLA  = 001142          490#    640*    648*   1295*   1395*
DISPRE  = 001174          454#    648
DRVADR  = 001320          586#    655
DRVBAR  = 001326          595#    775*    811     815     888*    921     925
DRVCSR  = 001330          596#    687*    705*    708     722*    723     738     741*    742     744     754*    755     776*
                          780     781*    783     788     792     793*    797     820*    825*    826     842*    843     859*
                          860     864     866*    868*    889     891*    895     896*    898     902     903*    907     950*
                          952*    953
DRVCTD  = 001334          601#    657    1015*   1023*
DRVCTR  = 001336          602#    661    1016*   1023    1024*
DRVDBR  = 001332          597#    655     704*    707     710     721*    753*    845     848*    862
DRVECT  = 001322          590#    658
DRVWCR  = 001324          594#    651     774*    801     803     887*    911     913
DSWR    = 177570          352#    489     639
DTI     = 006576          522     528     534     540     545     551     557     563     569     575     581    1639#
EMTVEC= 000030            441#    631*    632*
EMI     = 006056          520    1575#
EMI0    = 006354          561    1610#
EMI1    = 006400          567    1614#
```

DVDRBA.P11 CROSS REFERENCE TABLE

```
EM12    006430              573    1618#
EM13    006461              579    1623#
EM2     006120              526    1581#
EM3     006161              532    1587#
EM4     006214              538    1592#
EM5     006240              543    1596#
EM6     006270              549    1600#
EM7     006321              555    1605#
EOPT    003464              655     959#
EOPTA   003632              962     964     998    1003#
EOPTB   003640             1004#   1005
ERRVEC= 000004              434#    637     638#    649#   1383    1384#   1386#   1389#
GNS   = ******  U           453     678    1562    1563   1564    1565    1566    1568    1570    1571    1572
GTSWR = 104405              673    1568#
HT    = 000011              344#   1092    1133
ICOUNT  001346              609#    683#    963#    965#
IOTVEC= 000020              439#    629#    630#
LDBUF   003730              771    1046#
LF    = 000012              345#   1127    1133
MSTER   001344              608#    617#    619#    684    961
MTPS  = 106427              446#    680     768     777    881     892    1014    1025
MTST1   002012              701#   1006
MTST2   002134              737#
MTST3   002254              767#
PC    =%000007              365#    832#    959#    979#   982#    992#    997    1026#   1101#   1108#   1115#   1129#   1131#
                           1305#   1318#   1320#   1358#  1452#
PIRQ  = 177772              351#
PIROVE= 000240              445#
PR0   = 000000              368#
PR1   = 000040              369#
PR2   = 000100              370#
PR3   = 000140              371#
PR4   = 000200              372#
PR5   = 000240              373#
PR6   = 000300              374#
PR7   = 000340              375#
PS    = 177776              348#    349
PSW   = 177776              349#
PWRVEC= 000024              440#
RDCHR = 104407             1511    1571#
RDLIN = 104410             1572#
RESVEC= 000010              435#
RPRAM   001356              613#    887     888     891    916     917    928     929
RSTVEC  003672              832     959    1023#
R0    =%000000              356#    651#    653#    655    657#    659#   661     703#    704     706     715#    718#    719#
                            739#    741     750#    751    807#    808#   809#    810     917#    918#    919#    920     928#
                            932     943     944     989#   992    1034#  1037#   1046#   1051#   1084    1085#   1086    1089#
                           1223    1233#   1237    1253   1254    1267#  1330    1331#   1332#   1339#   1340#   1341#   1342#
                           1343#   1344    1349    1354#  1356#   1360   1362    1546    1547#   1548    1549#   1550#   1551#
                           1552#
R1    =%000001              357#    652#    653     654#   658#    659    660#    702#    716#    929#    934#    1035#   1036#
                           1047#   1050#   1224    1237#  1238    1242   1266#
R2    =%000002              358#    930#    932     936#   939#    940#   945     1048#   1050    1054#   1057#   1058#   1225
                           1236#   1240#   1243    1250#  1251#   1252   1257#   1265#
```

DVDRBA.P11 CROSS REFERENCE TABLE

```
SW6    = 000100       397#
SW7    = 000200       396#
SW8    = 000400       395#
SW9    = 001000       394#
TBITVE= 000014        436#
TIME     001340       606#   773*   778*   787*   790*   817*   818*   822*   823*   886*   893*   947*   948*
                     1003*  1004*
TKVEC  = 000060       443#
TPVEC  = 000064       444#
TRAPVE= 000034        442#   633*   634*
TRTVEC= 000014        437#
TYPDS  = 104407       987   1566#
TYPE   = 104400       667    985    988   1097   1193   1268   1298   1306   1329   1346   1348   1351   1353
                     1357   1364   1419   1420   1423   1436   1447   1466   1515   1518   1522   1562#
TYPOC  = 104401      1337   1361   1422   1563#
TYPON  = 104403      1565#
TYPOS  = 104402      1564#
XPRAM    001350       610#   774    775    776    806    807
$AUTOB   001134       486#   675*  1416   1537
$BDADR   001122       481#   707*   730*   783*   797*   803*   815*   898*   907*   913*   925*   944*  1639
$BDDAT   001126       483#   710*   711    744*   745    780*   792*   795    801*   811*   812*   813    895*
                      902*   905    911*   921*   922*   923    943*  1639
$BELL    001160       499#  1298   1314
$CHARC   004206      1099*  1109*  1116   1125*  1130#
$CKSWR   015226      1408#  1570
$CMTAG   001100       469#   622    623    631    635
$CM3   = 000000       499#
$CNTLG   015743      1419   1532#
$CNTLU   015756      1436   1531#
$CRLF    001165       501#  1098   1133   1306   1314   1329   1348   1353   1357   1447   1531
$DBLK    004424      1224   1268   1276#
$DOAGN   013406       981    990    996#
$DTBL    004644      1237   1272#
$ENDAD   013576       972#
$ENDCT   013544       968#
$ENDMG   013615       985   1000#
$ENULL   013612       988    999#
$EOP     013514       974#
$EOPCT   013536       980#   984
$ERFLG   001102       472#  1293*  1314   1373   1394*  1398
$ERMAX   001115       478#  1398
$ERROR   004464       631   1290#
$ERRPC   001116       479#  1300*  1301*  1302   1314   1335   1639
$ERRTB   001170       518#  1343
$ERRTY   005010      1318   1320#
$ERTTL   001112       476#  1299*  1314
$FILLC   001156       497#  1102   1133
$FILLS   001155       456#  1133
$GDADR   001120       480#
$GDDAT   001124       482#   706*   711    740*   745    749*   782*   794*   795    800*   806*   810*   813
                      897*   904*   905    910*   916*   920*   923    945*  1639
$GET42   002566       979#
$GTSWR   015336      1420#  1568
$HD    = 000003       325    326
```

```
$ICNT    001104        473#
$INTAG   001135        487#   1448   1537
$ITEMB   001114        477#   1302*  1314   1332
$LF      001166        502#   1133   1314   1522   1531
$LPADR   001106        474#   1393*  1396   1398
$LPERR   001110        475#
$MAIL  = ****** U      651    671    1086   1308   1393
$MNEW    006001        1423   1535#
$MSWR    005770        1420   1533#
$NULL    001154        495#   1104   1133
$OCNT    004434        1165*  1194*  1207#
$OMODE   004436        1160*  1164*  1169   1172*  1183*  1209#
$OVER    005222        1390   1395#
$PASS    001100        470#   977*   978*   986    999
$QUES    001164        500#   1133   1314   1466   1515   1531
$RDCHR   005520        1479#  1571
$RDDEC=  ****** U      1573
$RDLIN   005540        1507#  1572
$RDOCT=  ****** U      1573
$RDSZ  = 000010        1500#
$RTNAD   003610        998#
$R2A   = ****** U      1573
$SAVRE=  ****** U      1573
$SCOPE   005144        629    1377#
$SETUP=  000107        616#   628    629    631    633    635    667    668    976    1291   1311   1313   1378
                       1403   1537
$STUP  = 177777        616#
$SVLAD   005206        1387   1392#
$SWR   = 122000        325    326#   327#   333    334    335    336    499    635    971    977    991    997
                       999    1283   1284   1285   1286   1287   1296   1303   1308   1312   1314   1374   1380
                       1390   1392   1394   1398
$SWRMK=  000000        1374
$TKB     001146        492#   1401   1412   1429   1483   1489
$TKS     001144        491#   1401   1410   1426   1450*  1481   1487
$TN    = 000001        325#   328#
$TPB     001152        494#   1122*  1133
$TPFLG   001157        498#   1080   1133
$TPS     001150        493#   1120   1133
$TRAP    005012        633    1546#
$TRP   = 000011        1554#  1563#  1564#  1565#  1566#  1567#  1568   1569#  1570   1571#  1572#  1573#
$TRPAD   005034        1551   1561#
$TSTNM   001102        471#   976*   1295   1314   1373   1392*  1395   1398
$TTYIN   005746        1508   1509   1526   1530#
$TYPDN=  ****** U      1567
$TYPDS   004440        1222#  1566
$TYPE    002772        1080#  1554   1562
$TYPEC   004142        1101   1108   1115   1120#  1121   1452
$TYPEX   004210        1126   1128   1131#
$TYPOC   004226        1163#  1563
$TYPON   004232        1152   1165#  1565
$TYPOS   004212        1158#  1564
$XTSTR   005192        1381#
$$GET4=  000000        991#
$0FILL   004435        1159*  1163*  1173   1208#
```

```
$40CAT= ###### U          1305
.     = 006612             449#   453#   459#   468#   503    626    679#   999   1003   1133   1276#  1314   1367#
                          1398   1401   1530#  1531   1537   1638#
```

```
COMMEN      1#     446#
ENDCOM      1#     446#
ERROR     340#     713     747     784     798     804     816     899     908     914     926     946
ESCAPE      1#     446#
GETPRI      1#     446#
GETSWR      1#     446#    668#
MULT        1#     446#
NEWTST      1#     446#
OK        584#     711     745     795     813     905     923
POP         1#     446#   1263
PUSH        1#     446#   1222
REPORT      1#     446#
SCOPE     341#
SETPRI      1#     446#
SETTRA   1554#    1563    1564    1565    1566    1568    1570    1571    1572
SETUP       1#     446#    620
SKIP        1#     446#
SLASH       1#     446#
SPACE     446#
STARS       1#     446#    464     503     689     690     699     700     727     728     735     736     759     760     765
          766      835     836     839     840     853     854     856     857     871     872     878     879     968    1009
         1013     1019    1022    1028    1033    1041    1045    1065    1135    1212    1279    1314    1317    1323    1370
         1400     1403    1471    1500    1540    1641    1643
SWRSU       1#     446#    635#
TRNTRP   1554#
TYPBIN      1#     446#
TYPDEC      1#     446#    986
TYPNAM      1#     446#    663
TYPNUM      1#     446#
TYPOCS      1#     315#    446#
TYPOCT      1#     446#   1335    1359    1421
TYPTXT      1#     446#
$$CMRE    462#
$$CMTM    462#
$$ESCA      1#     446#
$$NEWT      1#     446#
$$SET    1554#    1563    1564    1565    1566    1568    1570    1571    1572
$$SKIP      1#     446#
.EQUAT      1#     315#    336
.HEADE      1#     315#
.KT11       1#
.SETUP      1#     315#    616
.SWRHI      1#     315#    329
.SWRLO    336#
.$ACT1      1#
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#     315#    447
.$CMTA      1#     315#    462
.$DB2D      1#
.$DB2O      1#
.$DIV       1#
```

DVDRBA.P11 CROSS REFERENCE TABLE

DVDRBA.P11 CROSS REFERENCE TABLE

	1192	1232	1235	1249	1252	1261	1302	1412	1429	1483	1489	1512	1517	1549	
NEG	808	918	1170	1231											
NOP	850	975	993	994	995										
RESET	682	991													
ROL	1176	1178	1179	1180	1182										
RTI	646	1091	1204	1271	1313	1397	1451	1499	1527						
RTS	1017	1026	1039	1053	1131	1320	1358	1552							
SUB	1238	1301													
TRAP	1554	1563	1564	1565	1566	1568	1570	1571	1572						
TST	669	684	961	1088	1096	1118	1187	1243	1253	1308	1362	1385	1443	1458	1548
TSTB	1080	1120	1245	1259	1410	1426	1481	1487							
.ASCII	500	501													
.ASCIZ	499	502	679	1000	1366	1531	1532	1533	1535	1575	1581	1587	1592	1596	1600
	1605	1610	1614	1618	1623	1627	1632								
.BLKB	1530														
.BLKW	1276														
.BYTE	471	472	477	478	486	487	495	496	497	498	999	1205	1206	1207	1208
	1528	1529													
.DSABL	1468														
.ENABL	1	315	1401												
.END	1645														
.ENDC	320	335	336	340	432	446	458	465	469	471	499	500	504	616	627
	628	631	633	635	651	667	671	677	679	690	691	700	701	728	729
	736	737	760	761	766	767	836	837	840	841	854	855	857	858	872
	873	879	880	969	970	971	973	976	981	984	985	989	991	997	999
	1000	1003	1010	1014	1020	1023	1029	1034	1042	1046	1066	1086	1136	1213	1280
	1283	1293	1300	1305	1306	1307	1308	1313	1314	1315	1318	1324	1339	1368	1371
	1374	1380	1391	1392	1393	1394	1398	1401	1402	1404	1432	1468	1472	1500	1501
	1508	1510	1513	1515	1531	1537	1541	1547	1550	1562	1563	1564	1565	1566	1567
	1568	1569	1570	1571	1572	1573	1642	1644							
.EQUIV	340	341	349	364	365	394	395	396	397	398	399	400	401	402	403
	422	423	424	425	426	427	428	429	430	431					
.EVEN	679	1367	1537	1638											
.IF	316	335	336	338	404	432	456	464	468	470	499	503	504	616	622
	627	629	631	633	635	651	666	667	668	671	678	690	691	699	700
	727	728	735	736	759	760	765	766	835	836	839	840	841	854	856
	857	871	872	878	879	968	969	970	971	972	973	975	980	983	985
	989	991	997	999	1000	1009	1013	1019	1022	1028	1033	1041	1045	1065	1086
	1135	1212	1279	1282	1292	1296	1303	1305	1306	1308	1312	1313	1314	1317	1323
	1338	1354	1370	1373	1374	1379	1380	1390	1392	1393	1394	1397	1398	1400	1402
	1403	1404	1432	1471	1472	1500	1508	1509	1513	1514	1530	1531	1537	1540	1546
	1550	1554	1563	1564	1565	1566	1567	1568	1570	1571	1572	1573	1641	1643	
.IFF	335	336	338	465	468	470	499	504	627	666	667	690	691	700	701
	728	729	736	737	760	761	766	767	836	837	840	841	854	855	857
	858	872	873	879	880	969	972	975	981	984	999	1010	1014	1020	1023
	1029	1034	1042	1046	1066	1136	1213	1280	1282	1296	1312	1313	1314	1315	1318
	1324	1339	1368	1371	1390	1392	1398	1401	1404	1472	1474	1479	1500	1501	1510
	1514	1531	1541	1547	1642	1644									
.IFT	679	1306	1392	1474	1479										
.IFTF	679	1305	1392	1419	1472	1475									
.IIF	315	320	325	326	333	334	335	453	503	628	631	635	667	970	976
	977	987	999	1003	1133	1283	1284	1285	1286	1287	1291	1311	1313	1314	1336
	1361	1374	1378	1394	1398	1401	1422	1523	1531	1537	1562	1563	1564	1565	1566
	1568	1570	1571	1572											

DVDRBA.P11 CROSS REFERENCE TABLE

ERRORS DETECTED: 0

#,DVDRBA.SEQ/SOL/CRF/ML:TOC/DS:ERFZ=DVDRBA.SML,DVDRBA.P11
RUN-TIME: 40 45 3 SECONDS
CORE USED: 32K

JO4

Spooler running 4 Seconds, 31 VTS, 177 disk reads, 3 disk writes, 47 pages