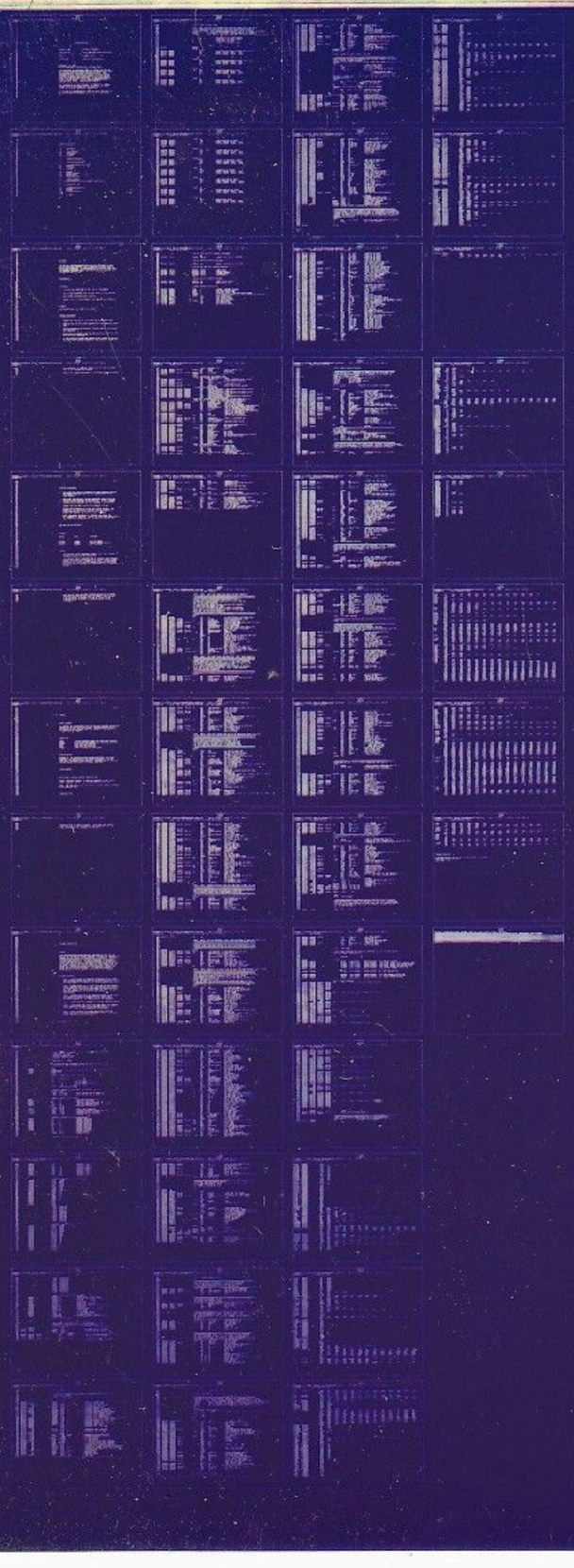
DRV11B

INTERPROCESSOR EXERCISER

MD-11-DVDRB-A

EP-DVDRB-A-DL-A NOV 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

TABLE OF CONTENTS

1.0 ABSTRACT

2.0 REQUIREMENTS
2.1 EQUIPMENT
2.2 STORAGE

3.0 LOADING PROCEDURE

4.0 STARTING PROCEDURE

5.0 SOFTWARE SWITCH REGISTER
5.1 OPTIONS
5.2 CONTROL

6.0 ERRORS
6.1 ERROR COMMENT
6.2 ERROR DATA
6.3 ERROR RECOVERY

7.0 MISCELLANEOUS
7.1 DRV11B LDS & VECTOR ADDRESS MODIFICATION

8.0 EXECUTION TIME

9.0 PROGRAM DESCRIPTION
9.1 GENERAL
9.2 PROGRAM SEGMENTS

10.0 LISTING

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 ARE SET WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1. IF THEY ARE NOT SET TO THESE LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' OUT 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 SLAVE STARTED AS MASTER WILL REPORT THE 'END OF PASS' MESSAGE).

5. START THE SLAVE **FIRST** AT ADDRESS 200 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 L[illegible]E DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER [illegible]FICATION OF THE ERROR CAN BE OBTAINED FROM THE COMMENT [illegible] 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 CC[illegible]ND/STATUS REGISTER AND COMPARE IT TO THE CODE WRITTEN.

3. MTST3 - MASTER [illegible]S OUT A 32 WORD DATA BLOCK TO THE SLAVE AND CHECKS F[illegible]ER INTE[illegible]UPT STATUS, WORD COUNT, AND BUFFER AD[illegible]RESS 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.
                                ;*
000001                          $TN=1
160000                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
122000                          $SWR=122000
000001                          $TN=1
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          ----------------
                                ;*        15            HALT ON ERROR
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        10            BELL ON ERROR
                                .SETTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQU.V  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                          SWR=    177570          ;;HARDWARE SWITCH REGISTER
177570                          DISPLY= 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
        [illegible]     BIT10=  2000
        001000          BIT09=  1000
        000400          BIT08=  400
        000200          BIT07=  200
        000100          BIT06=  100
        [illegible]     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

000000                                  .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
000174                                  .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001364                  JMP     @#START1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001372                  JMP     @#START2        ;GO START AS SLAVE COMPUTER
000100                                  .=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                  $CNTAG:                         ;;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 SGOADR  SGODAT  SBODAT
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 SGOADR  SGODAT  SBODAT
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 SGOADR  SGODAT  SBODAT
001216  000000                          0

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

```
                                        ;ERROR  5
001230  006240                                  EM5             ;WORD COUNT ER ON 'DATI'
001232  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001234  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001236  000000                                  0

                                        ;ERROR  6
001240  006270                                  EM6             ;BUFFER ADRS ER ON 'DATI'
001242  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001244  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001246  000000                                  0

                                        ;ERROR  7
001250  006321                                  EM7             ;FAILED TO INTR ON A 'DATO'
001252  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001254  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001256  000000                                  0

                                        ;ERROR  10
001260  006354                                  EM10            ;STATUS ER ON 'DATO'
001262  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001264  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001266  000000                                  0

                                        ;ERROR  11
001270  006400                                  EM11            ;WORD COUNT ER ON 'DATO'
001272  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001274  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001276  000000                                  0

                                        ;ERROR  12
001300  006430                                  EM12            ;BUFFER ADRS ER ON 'DATO'
001302  006503                                  DH1             ;ERRPC  BUSADR  EXPCT   RCVD
001304  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
001306  000000                                  0

                                        ;ERROR  13
001310  006461                                  EM13            ;DATA ER ON 'DATO'
001312  006540                                  DH2             ;ERRPC  MEMADR  EXPCT   RCVD
001314  006576                                  DT1             ;$ERRPC $BDADR  $GDDAT  $BDDAT
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                  DRVBUF: 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     MSTER           ;THIS IT MASTER START
001370  000403                  START:  BR      START           ;SKIP NEXT
001372  012737  177777  001344  START2: MOV     #-1,MSTER       ;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                  1$:     CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001406  022706  001140                  CMP     #SWR,R6 ;;DONE?
001412  001374                          BNE     1$              ;;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  005264  000030          MOV     #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001442  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001450  012737  006012  000034          MOV     #SYSTRP,@#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          1$:     MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001470  012737  001524  000004          MOV     #2$,@#ERRVEC    ;;SET UP ERROR VECTOR
001476  012737  177570  001140          MOV     #SWR,@#SWREG    ;;SETUP FOR A HARDWARE SWITCH REGISTER
001504  012737  177570  001142          MOV     @#SWREG,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          2$:     MOV     #65$,(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     #DRVWDR,R0      ;SET UP REG ADRS POINTERS
001556  012701  001320                  MOV     DRVA ,R1        ;GET BASE ADRS
001562  010120                  SETUP2: MOV     R1,(R0)+        ;LOAD EM
001564  062701  000002                  ADD     #2,R1           ;
001570  022700  001334                  CMP     #DRVCSR+2,R0    ;ALL DONE?
001574  001372                          BNE     SETUP2          ;BR IF NOT
001576  012700  001234                  MOV     #DRVCTO,R0      ;SET UP DRV11B VECTOR ADRS POINTER
001602  012701  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?
```

```
MAINDEC-11-DVDRB-A  DRV11B INTERPROCESSOR EXERCISER     MACY11 27(665)  12-OCT-76  13:23  PAGE 18
DVDRBA.P11      GET VALUE FOR SOFTWARE SWITCH REGISTER

   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  104435                          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 F SET 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$:     CLR     R1              ;R1 SAYS FLOAT 0 LEFT WHEN ZERO
002014  012700  177776                          MOV     #-2,R0          ;START WITH #177776 IN R0
002020  010077  177306                  2$:     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 EITHER 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  017[illegible] 177134  001126          MOV     @DRVCSR,$BDDAT  ;READ THE CSR
002176  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
[illegible]  001402                          BEQ     3$              ;BR IF SO
[illegible]  104402                          ERROR   2               ;SLAVE FAILED TO ECHO 'STAT' BITS
[illegible]  000761                          BR      1$              ;LOOP ON ERROR ALWAYS
[illegible]  062737  001002  001124  3$:     ADD     #1002,$GDDAT    ;ADVANCE EXPECTED
[illegible]  062700  000002                  ADD     #2,R0           ;ADVANCE PTRN
[illegible]  032700  000020                  BIT     #20,R0          ;ALL BEEN DONE?
[illegible]  001751                          BEQ     1$              ;BR IF NOT
[illegible]  012777  177002  177072          MOV     #177002,@DRVDBR ;TELL SLAVE TO GO TO NEXT TEST
[illegible]  005077  177064                  CLR     @DRVCSR         ;SET THERE CYCLE
[illegible]  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  001350  177014          MOV     XP[illegible],@DRVWCR   ;SET UP WORD COUNT
002310  013777  001352  177010          MOV     XP[illegible],@DRVBAR   ;SET UP BUFFER ADRS
002316  013777  001354  177004          MOV     XP[illegible],@DRVCSR   ;SET UP CONTROL - IE, FNCT 3 & GO(CLRS CYCLE)
002324  106427  000000                  MTPS,   0               ;ALLOW THE RDY INTR
[illegible]  005337  001340          2$:     DEC     TIME            ;WAIT FOR INTR
[illegible]  001375                          BNE     2$              ;UNTIL 0
[illegible]  017737  176756  001126          MOV     @DRVCSR,$BDDAT  ;READ CSR - SHOULD NEVER GET HERE
[illegible]  042777  000100  176756          BIC     #100,@DRVCSR    ;DISABLE IE
[illegible]  012737  005710  001124          MOV     #5710,$GDDAT    ;LD EXPECTED - STAT A & C, CYCLE, RDY, IE, FNCT 3
[illegible]  013737  001330  001122          MOV     DRVCSR,$BDRDR   ;SET UP CSR ADRS
[illegible]  104003                          ERROR   3               ;DATI FAILED TO CAUSE A WC INTR
[illegible]  000477                          BR      6$              ;GO RESYNC ON ERROR
[illegible]  [illegible]26                  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
00[illegible]  013700  001350            MOV     XPRAM,R0        ;GET WC
00[illegible]  005400                    NEG     R0              ;GET ACTUAL #
00[illegible]  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  000620                           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:
002736                  1$:     BIC     #400,@#RVCSR    ;TELL MASTER WE ARE READY
002736  042777  000400  176364
002744  032777  000400  176356  2$:     BIT     #400,@#RVCSR    ;LOOK FOR CYCLE
002752  001774                          BEQ     2$              ;WAIT ON IT
002754  022777  177002  176350          CMP     #177002,@#RVDBR ;TEST TERMINATOR?
002762  001412                          BEQ     3$              ;[?] IF SO
002764  017737  176340  001342          MOV     @#RVCSR,SAVE    ;READ THE CSR
002772  042737  170777  001342          BIC     #170777,SAVE    ;SAVE ONLY THE STAT BITS
003000  113777  001343  176322          MOVB    SAVE+1,@#RVCSR  ;ECHO WITH FNCT BITS
003006  000753                          BR      1$              ;LOOK FOR NEXT 'STAT' CODE
003010  005077  176314          3$:     CLR     @#RVCSR         ;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:
000014                  1$:     MTPS,   200             ;NO INTRS ALLOWED YET
000014  106427  000200
000020  004537  003652          JSR     R5,SETVEC       ;SET UP THE INTR RETURN ADRS
000024  003142                  3$                      ;RETURN TO 3$ ON DATO INTR
000026  004537  003712          JSR     R5,CLRBUF       ;GO CLR 'DBUF'
000032  177740                  -32.                    ;DO 32. LOCATIONS
000034  012737  100000  001340  MOV     #100000,TIME    ;SET UP A TIMER VALUE
000042  013777  001356  176254  MOV     #1,@#RVCSR      ;SET UP WORD COUNT
000050  013777  001360  176250  MOV     #BUF,@#RVBAR    ;SET UP BUFFER ADRS
000056  042777  000400  176244  2$:     BIT     #400,@#RVCSR    ;LOOK FOR CYCLE
000064  001774                  BEQ     2$              ;WAIT FOR IT
000066  013777  001362  176234  MOV     #BUF+4+4,@#RVCSR ;SET UP CONTROL - IE, FNCT 3 & 1 & GO
000074  106427  000000          MTPS,   0               ;ALLOW THE INTR
000100  005337  001340          4$:     DEC     TIME    ;WAIT FOR INTR
000104  001375                  BNE     4$              ;UNTIL ZERO
000106  017737  176216  001126  MOV     @#RVCSR,$RODAT  ;READ CSR - SHOULD NEVER GET HERE
000114  042777  000100  176206  BIC     #100,@#RVCSR    ;DISABLE IE
000122  012737  004312  001124  MOV     #4312,$SODAT    ;LOOKED EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
000130  013737  001330  001122  MOV     DRVCSR,$DVADR   ;SET UP CSR ADRS
000136  104007                  ERROR   7               ;DATO FAILED TO CAUSE A WC INTR
000140  000517                  BR      10$             ;GO RESYNC ON ER
000142  022626          3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RETURN
000144  017737  176160  001126  MOV     @#RVCSR,$RODAT  ;READ STATUS
000152  042777  000100  176150  BIC     #100,@#RVCSR    ;DISABLE IE
```

```
 904  003160  012737  004312  001124          MOV     #4312,$GDDAT    ;LD EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
 905  003166  023737  001124  001126          CMP     $GDDAT,$$DAT    ;CORRECT?
 906  003174  001405                          BEQ     4$              ;BR IF SO
 907  003176  013737  001330  001122          MOV     DRVCSR,$BOADR   ;SET UP CSR ADRS
 908  003204  104010                          ERROR   10              ;DATO STATUS ERROR
 909  003206  000474                          BR      10$             ;GO RESYNC ON ERROR
 910  003210  005037  001124          4$:     CLR     $GDDAT          ;LD EXPECTED WC
 911  003214  017737  176104  001126          MOV     @DRVWCR,$BDDAT  ;READ WCR
 912  003222  001405                          BEQ     5$              ;BR IF SO
 913  003224  013737  001324  001122          MOV     DRVWCR,$BOADR   ;SET UP WCR ADRS
 914  003232  104011                          ERROR   11              ;DATO WORD COUNT ERROR
 915  003234  000461                          BR      10$             ;GO RESYNC ON ERROR
 916  003236  013737  001360  001124  5$:     MOV     RPRAM+2,$GDDAT  ;GET STARTING ADRS OF XFER
 917  003244  013700  001356                  MOV     RPRAM,R0        ;GET WC
 918  003250  005400                          NEG     R0              ;GET ACTUAL #
 919  003252  006300                          ASL     R0              ;CONVERT TO WORD
 920  003254  060037  001124                  ADD     R0,$GDDAT       ;ADD TO BASE ADRS
 921  003260  017737  176042  001126          MOV     @DRVBAR,$BDDAT  ;READ BAR ADRS
 922  003266  042737  000001  001126          BIC     #BIT00,$BDDAT   ;ELIMINATE LSB
 923  003274  023737  001124  001126          CMP     $GDDAT,$$DAT    ;CORRECT?
 924  003302  001405                          BEQ     6$              ;BR IF SO
 925  003304  013737  001326  001122          MOV     DRVBAR,$BOADR   ;SET UP BAR ADRS
 926  003312  104012                          ERROR   12              ;DATO BUFFER ADRS ERROR
 927  003314  000431                          BR      10$             ;GO RESYNC ON ERROR
 928  003316  013700  001360          6$:     MOV     RPRAM+2,R0      ;GET BUFFER ADRS
 929  003322  013701  001356                  MOV     RPRAM,R1        ;GET WORD COUNT
 930  003326  012702  177776          7$:     MOV     #177776,R2      ;GET 1ST DATA PTRN (FLOATING 0)
 931  003332  005003                          CLR     R3              ;R3 KEYS WHEN TO SHIFT PTRN
 932  003334  020220                  8$:     CMP     R2,(R0)+        ;COMPARE DATA IN DBUF TO EXPECTED
 933  003336  001011                          BNE     9$              ;BR IF DATA ERROR
 934  003340  005201                          INC     R1              ;COUNT THE WORD COUNT
 935  003342  001431                          BEQ     13$             ;BR IF DATA CHECKS DONE
 936  003344  005102                          COM     R2              ;CONVERT PTRN TO FLOATING 1
 937  003346  005103                          COM     R3              ;TIME TO SHIFT?
 938  003350  001371                          BNE     8$              ;BR IF NOT - GO CK NEXT
 939  003352  006302                          ASL     R2              ;FLOAT PTRN LEFT
 940  003354  005202                          INC     R2              ;KEEP LSB SET
 941  003356  103363                          BCC     7$              ;GO RESET FLOATING PTRN
 942  003360  000765                          BR      8$              ;GO CHECK NEXT
 943  003362  014037  001126          9$:     MOV     -(R0),$BDDAT    ;GET BAD DATA
 944  003366  010037  001122                  MOV     R0,$BOADR       ;GET MEM ADRS OF DATA ER
 945  003372  010237  001124                  MOV     R2,$GDDAT       ;LD EXPECTED DATA
 946  003376  104013                          ERROR   13              ;DATO DATA ERROR
 947  003400  012737  000100  001340  10$:     MOV     #100,TIME       ;LET MASTER FINISH ITS CHECKS
 948  003406  005337  001340          11$:    DEC     TIME            ;COUNT AWAY
 949  003412  001375                  12$:    BNE     11$             ;UNTIL DONE
 950  003414  112777  000002  175706          MOVB    #2,@DRVCSR      ;TELL MASTER WE HAVE AN ERROR
 951  003422  000137  003330                  JMP     1$              ;DO TEST AGAIN ON ERROR
 952  003426  105077  175676          13$:    CLRB    @DRVCSR         ;TELL MASTER ALL OK
 953  003432  017737  175672  001342  14$:    MOV     @DRVCSR,SAVE    ;READ CSR
 954  003440  042737  170777  001342          BIC     #170777,SAVE    ;SAVE 'STAT' BITS ONLY
 955  003446  001405                          BEQ     EOPT            ;BR IF MASTER DONE WITH NO ER'S
 956  003450  022737  001000  001342          CMP     #1000,SAVE      ;DOES MASTER HAVE AN ERROR?
 957  003456  001365                          BNE     14$             ;BR IF NOT
```

```
 958  003460  000137  003014            JMP     1$                      ;YES - DO TEST AGAIN ON ERROR
 959  003464  004737  003672  EOPT:     JSR     PC,RSTVEC               ;GO RESTORE VECTOR
 960  003470  104406                    CKSWR                           ;GO CHECK SWR
 961  003472  005737  001344            TST     MSTER                   ;WERE WE STARTED AS MASTER?
 962  003476  001055                    BNE     EOPTA                   ;BR IF NOT
 963  003500  005337  001346            DEC     ICOUNT                  ;COUNT TEST PASS
 964  003504  001052                    BNE     EOPTA                   ;BR IF NOT DUE FOR 'END PASS' MSG
 965  003506  012737  003000  001346    MOV     #3000,ICOUNT            ;RESET PASS COUNT
 966                      .SBTTL  END OF PASS ROUTINE
 967
 968                      ;;**********************************************************************
 969                      ;*INCREMENT THE PASS NUMBER ($PASS)
 970                      ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 971                      ;*IF THERE IS A MONITOR GO TO IT
 972                      ;*IF THERE ISN'T JUMP TO EOPTA
 973
 974  003514              SEOP:
 975  003514  000240                    NOP
 976  003516  005037  001102            CLR     $TSTNM                  ;;ZERO THE TEST NUMBER
 977  003522  005237  001100            INC     $PASS                   ;;INCREMENT THE PASS NUMBER
 978  003526  042737  100000  001100    BIC     #100000,$PASS           ;;DON'T ALLOW A NEG. NUMBER
 979  003534  005327                    DEC     (PC)+                   ;;LOOP?
 980  003536  000001          SEOPCT:   .WORD   1
 981  003540  003022                    BGT     $DOAGN                  ;;YES
 982  003542  012737                    MOV     (PC)+,@(PC)+            ;;RESTORE COUNTER
 983  003544  000001          SENDCT:   .WORD   1
 984  003546  003536                    $EOPCT
 985  003550  104400  003615            TYPE    ,SENMSG                 ;;TYPE "END PASS #"
 986  003554  013746  001100            MOV     $PASS,-(SP)             ;;SAVE $PASS FOR TYPEOUT
 987  003560  104404                    TYPOS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
 988  003562  104400  003612            TYPE    ,SENULL                 ;;TYPE A NULL CHARACTER
 989  003566  013700  000042  SGET42:   MOV     @#42,R0                 ;;GET MONITOR ADDRESS
 990  003572  001405                    BEQ     $DOAGN                  ;;BRANCH IF NO MONITOR
 991  003574  000005                    RESET                           ;;CLEAR THE WORLD
 992  003576  004710          SENDAD:   JSR     PC,(R0)                 ;;GO TO MONITOR
 993  003600  000240                    NOP                             ;;SAVE ROOM
 994  003602  000240                    NOP                             ;;FOR
 995  003604  000240                    NOP                             ;;ACT11
 996  003606              $DOAGN:
 997  003606  000137                    JMP     @(PC)+                  ;;RETURN
 998  003610  003632          $RTNAD:   .WORD   EOPTA
 999  003612      377     377     000  SENULL:   .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
1000  003615      015  042412  042116  SENMSG:   .ASCIZ  <15><12>/END PASS #/
1001  003622  050040  051501  020123
1002  003630  000043
1003  003632  012737  001000  001340  EOPTA:    MOV     #1000,TIME      ;SET UP A COUNT
1004  003640  005337  001340  EOPTB:    DEC     TIME                    ;STALL FOR OTHER CPU TO BECOME SLAVE
1005  003644  001375                    BNE     EOPTB                   ;UNTIL TIME=0
1006  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
                                ;;*****************************************************************
003452  106427  000200          SETVEC: MTPS,   200             ;SET UP FOR NO INTERRUPT
003456  012577  175452                  MOV     (R5)+,@DRVCT0   ;SET UP INTR RETURN ADRS
003462  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$:     COM     R2              ;CONVERT PTRN TO FLOATING 1
003756  005103                          COM     R3              ;SHOULD WE SHIFT?
003760  001371                          BEQ     2$              ;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          $TYPE:  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                  6O$:    MOV     (SP)+,R0        ;;RESTORE R0
004022  062716  000002          3$:     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  105067  004206                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
004056  000765                          BR      2$              ;;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
```

```
                                95:     JSR     PC,$TYPEC       ;;TYPE A SPACE
                                        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
                                        BNE     9$              ;;TAB STOP
                                        TST     (SP)+           ;;POP SPACE OFF STACK
                                        BR      2$              ;;GET NEXT CHARACTER
                                $TYPEC: TSTB    @#TPS           ;;WAIT UNTIL PRINTER IS READY
                                        BPL     $TYPEC
                                        MOVB    2(SP),@#TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
                                        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
                                        BNE     1$              ;;BRANCH IF NO
                                        CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
                                        BR      $TYPEX          ;;EXIT
                                1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
                                        BEQ     $TYPEX          ;;BRANCH IF YES
                                        INCB    (PC)+           ;;COUNT THE CHARACTER
                                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
                                $TYPEX: RTS     PC

                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON--ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC--ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT

                                $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
                                        MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
                                        MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
                                        ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                                        BR      $TYPON
                                $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
                                        MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
                                $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                                        MOV     R3,-(SP)        ;;SAVE R3
                                        MOV     R4,-(SP)        ;;SAVE R4
                                        MOV     R5,-(SP)        ;;SAVE R5
```

```
                                        MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                                        NEG     R4
                                        ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
                                        MOVB    R4,$OMODE       ;;SAVE IT FOR USE
                                        MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
                                        MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
                                        CLR     R3              ;;CLEAR THE OUTPUT WORD
                                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
                                        BR      3$              ;;GO DO MSB
                                2$:     ROL     R5              ;;FORM THIS DIGIT
                                        ROL     R5
                                        ROL     R5
                                        MOV     R5,R3
                                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
                                        DECB    $OMODE          ;;TYPE THIS DIGIT?
                                        BPL     7$              ;;BR IF NO
                                        BIC     #177770,R3      ;;GET RID OF JUNK
                                        BNE     4$              ;;TEST FOR 0
                                        TST     R4              ;;SUPPRESS THIS 0?
                                        BEQ     5$              ;;BR IF YES
                                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
                                        BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
                                5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
                                        MOVB    R3,B$           ;;SAVE FOR TYPING
                                        TYPE    ,B$             ;;GO TYPE THIS DIGIT
                                7$:     DECB    $OCNT           ;;COUNT BY 1
                                        BGT     2$              ;;BR IF MORE TO DO
                                        BLT     6$              ;;BR IF DONE
                                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
                                        BR      2$              ;;GO DO THE LAST DIGIT
                                6$:     MOV     (SP)+,R5        ;;RESTORE R5
                                        MOV     (SP)+,R4        ;;RESTORE R4
                                        MOV     (SP)+,R3        ;;RESTORE R3
000002  000004                          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                                        MOV     (SP)+,(SP)
                                        RTI                     ;;RETURN
                                B$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
000000                          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE


                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CONVERT 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
                                ;*IN PLACE 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
                                ;*      TYPDS                   ;;GO TO THE ROUTINE

004440                          $TYPDS:
```

```
                                                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$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
                    004654                      MOV     #SDBLK,R3       ;;SETUP THE OUTPUT POINTER
                    000040                      MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
                                        2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
                    004644                      MOV     SDTBL(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$:     ADC     R1,R5           ;;ADD BACK THE CONSTANT
                                                TST     R2              ;;CHECK IF BCD DIGIT=0
                                                BNE     5$              ;;FALL THROUGH IF 0
                                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
                                                BMI     7$              ;;BR IF YES
                                        5$:     ASLB    (SP)            ;;0?
                                                BCC     6$              ;;BR IF NO
                    177777  000001              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 BLANK 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    ,SDBLK          ;;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.
                                    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

004664                          $ERROR:
004764  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
004766  104406                          CKSWR           ;GO LOOK FOR SWR CHANGE
004770  005237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
004774  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
004776  013777  001102  174236          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
005004  013777  002000  174226          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
005012  001402                          BEQ     1$              ;;NO - SKIP
005014  104400  001160                  TYPE    ,$BELL          ;;RING BELL
005020  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
005024  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
005030  162737  000002  001116          SUB     #2,$ERRPC
005036  117737  174154  001114          MOVB    @2(SP),$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
005044  032777  020000  174166          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
005052  001004                          BNE     20$             ;;SKIP TYPEOUTS
005054  004737  005000                  JSR     PC,$WRCK        ;;GO TO USER ERROR ROUTINE
005060  104400  001165                  TYPE    ,$CRLF
005064                          20$:
005064  005777  174150          2$:     TST     @SWR            ;;HALT ON ERROR
005070  100002                          BPL     3$              ;;SKIP IF CONTINUE
005072  000000                          HALT                    ;;HALT ON ERROR!
005074  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
005076                          3$:
005076  000002                          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    @#$ITEMB,R0
1333  005024  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
1334                                                                  ;;TYPE THE PC OF THE ERROR
1335  005026  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
1336                                                                  ;;ERROR ADDRESS
1337  005032  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1338  005034  000[illegible]                  BR      6$              ;;GET OUT
1339  005036  005[illegible]          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
1340  005040  006[illegible]                  ASL     R0              ;;       WORK FOR THE ERROR TABLE
1341  005042  006[illegible]                  ASL     R0
                                              ASL     R0
                001170                        ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
                005062                        MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
                0[illegible]                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
                1[illegible]                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
                [illegible]             2$:   .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
                001165                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                012037  005100          3$:   MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
                001[illegible]                BEQ     5$              ;;SKIP TYPEOUT IF 0
                104[illegible]                TYPE                    ;;TYPE THE "DATA HEADER"
                0[illegible]            4$:   .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
                001165                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                011000                  5$:   MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
                001004                        BNE     4$              ;;GO TYPE THE DATA
                012[illegible]          6$:   MOV     (SP)+,R0        ;;RESTORE R0
                10[illegible]           001165  TYPE  $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                000207                        RTS     PC              ;;RETURN
                                        7$:
                013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                104401                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                005710                        TST     (R0)            ;;IS THERE ANOTHER ML   ??
                001770                        BEQ     6$              ;;IF NO
                104400  005140                TYPE    ,8$             ;;TYPE TWO(2) SPACES
                000774                        BR      7$              ;;LOOP
1366  005140  020040      000         8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
1367            005144                        .EVEN
1368                                  .SBTTL  SCOPE HANDLER ROUTINE
1369
1370                                  ;;**************************************************************
1371                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1372                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1373                                  ;*AND LOAD THE ERROR FLAG ($ERRFLG) INTO DISPLAY<15:08>
1374                                  ;*CALL
1375                                  ;*      SCOPE           ;;SCOPE=IOT
1376
1377  005144                          $SCOPE:
1378  005144  104406                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
1379  005146  104406                          CKSWR                   ;;GO LOOK FOR SWR CHANGE
1380                                  ;#####START OF CODE FOR THE XOR TESTER#####
1381  005150  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1382                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1383  005152  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1384  005156  012737  005176  000004          MOV     #$$,@#ERRVEC    ;;SET FOR TIMEOUT
```

```
005164  005737  177060                  TST     @#177060                ;;TIME OUT ON XOR?
0[illegible]5170  01[illegible]37  000004         MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
0[illegible]5174  0[illegible]4                         BR      $SVLAD                  ;;GO TO THE NEXT TEST
005176  022626                  5$:     CMP     (SP)+,(SP)+             ;;CLEAR THE STACK AFTER A TIME OUT
0[illegible]  012637  000004                  MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
0[illegible]4  000406                           BR      $OVER                   ;;LOOP ON THE PRESENT TEST
00[illegible]06                          6$:;#####END OF CODE FOR THE XOR TESTER#####
005206  105237  001102          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
005212  011637  001106                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
00[illegible]16  105037  001103                  CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
00[illegible]22  013777  001102  173712  $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
005230  013716  001106                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
005234  000002                          RTI                             ;;FIXES PS
                                .SBTTL  TTY INPUT ROUTINE

                                ;;*****************************************************************
                                .ENABL  LSB

                                ;;*****************************************************************
                                ;*SOFTW[illegible]E SWITCH REGISTER CHANGE ROUTINE.
                                ;*[illegible]UTINE IS ENTERED FR[illegible] THE TRAP HA[illegible]DLER, AND WILL
                                ;*[illegible]RVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*[illegible]N OPERATING IN TTY FLAG MODE.
00[illegible]36  0[illegible]2737  000176  001140  $CKSWR: CMP     #[illegible]REG,SWR             ;;IS THE SOFT-SWR SELECTED?
0[illegible]  [illegible]074                          [illegible]      1[illegible]                     ;;[illegible]ANCH IF NO
[illegible]6  1[illegible]777  173672                  TSTB    @$TKS                   ;;CHAR THERE?
0[illegible]2  100071                          BPL     15$                     ;;IF NO, DON'T WAIT AROUND
[illegible]  117746  173[illegible]6                  MOVB    @$TKB,-(SP)             ;;SAVE THE CHAR
[illegible]  [illegible]716  17[illegible]                  BIC     #^C177,(SP)             ;;STRIP-OFF THE ASCII
0[illegible]  [illegible]726  000007                  CMP     #7,(SP)+                ;;IS IT A CONTROL G?
0[illegible]  [illegible]2                          BNE     15$                     ;;NO, RETURN TO USER
0[illegible]2  123727  001134  000001          CMPB    $AUTOB,#1               ;;ARE WE RUNNING IN AUTO-MODE?
005300  001456                          BEQ     15$                     ;;BRANCH IF YES

005302  104400  005763                  TYPE    ,$CNTLG                 ;;ECHO THE CONTROL-G (^G)
005306  104400  005770          $GTSWR: TYPE    ,$MSWR                  ;;TYPE CURRENT CONTENTS
005312  013746  000176                  MOV     SWREG,-(SP)             ;;SAVE S[illegible]EG FOR TYPEOUT
00[illegible]316  104401                          TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
00[illegible]  104400  006001                  TYPE    ,$MNEW                  ;;PROMPT FOR NEW SWR
00[illegible]4  005046                  19$:    CLR     -(SP)                   ;;CLEAR COUNTER
005326  0[illegible]046                          CLR     -(SP)                   ;;THE NEW SWR
005330  105777  173610          7$:     TSTB    @$TKS                   ;;CHAR THERE?
005334  100375                          BPL     7$                      ;;IF NOT TRY AGAIN

005336  117746  173604                  MOVB    @$TKB,-(SP)             ;;PICK UP CHAR
005342  042716  177600                  BIC     #^C177,(SP)             ;;MAKE IT 7-BIT ASCII



005346  021627  000025          9$:     CMP     (SP),#25                ;;IS IT A CONTROL-U?
0[illegible]2  001005                          BNE     10$                     ;;BRANCH IF NOT
0[illegible]4  104400  005756                  TYPE    ,$CNTLU                 ;;YES, ECHO CONTROL-U (^U)
00[illegible]60  062706  000006          20$:    ADD     #6,SP                   ;;IGNORE PREVIOUS INPUT
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  005764  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$:    ADD     #6,SP           ;;CLEAR UP STACK
1447  005414  104430  001165          14$:    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,@#TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1451  005436  000002          15$:    RTI                             ;;RETURN
1452  005440  004737  004142          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1453  005444  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
1454  005450  002430                          BLT     18$             ;;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$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1464  005504  050616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1465  005510  000707                          BR      7$              ;;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     2$              ;;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  026627  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  001006                        BNE     2$                      ;;LOOP IF NOT RETURN
1521  005712  105013                        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  000000                9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
1529  005745  000                           .BYTE   0                       ;;TERMINATOR
1530  005746  000000                $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
1531  005756  025536  005015        $CNTLU: .ASCIZ  /^U/<15><12>            ;;CONTROL "U"
1532  005763  136 107  000012        $CNTLG: .ASCIZ  /^G/<15><12>            ;;CONTROL "G"
1533  005770  015 123  020122        $SWR:   .ASCIZ  <15><12>/SWR = /
1534  005776  020075                $
1535  006001  040 040  047040  053505  $NEW:  .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
```

```
006014  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
006020  005740                  TST     -(R0)           ;;BACKUP BY 2
006022  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
006024  006300                  ASL     R0              ;;POSITION FOR INDEXING
006026  016000  006034          MOV     $TRPAD(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 NU  R (WITH LEADING ZEROS)
006040  004212                  $TYPOS  ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NU  R (NO LEADING ZEROS)
006042  004252                  $TYPON  ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUM R (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
0     2 0     0                 $  CHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
0  054  0    0                  $  LIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE

                        .SBTTL  ASCII MESSAGES
0   6   0   123 0 3101          EM1:    .ASCIZ  /SLAVE FAILED TO ECHO DBR CONTENTS/
     4  0     5 0 6111  0
     2        0  0117   0      5
       0  0   510     40   0   2
   106  0414     117   0   24
   114      116      123
        0         101  0 105    EM2:    .ASCIZ  /SLAVE FAILED TO ECHO 'STAT' BITS/
                  111
0  134  0          17   0     5
0  142  0   510    40         3
0       0    101   17   0     2
0       0 0510  4     00
0  161       106  0     1 042514    EM3:    .ASCIZ  /FAILED TO INTR ON A 'DATI'/
0  166 0 0104   0     24 044440
0  174        116       2 047117
0     2                 5 040  4
0     0    0     4       47
0  214         23        01 051525  EM4:    .ASCIZ  /STATUS ER ON 'DATI'/
0     2        0         2  047117
0              0   0    4   044524
0               7
0             27  042122  041440    EM5:    .ASCIZ  /WORD COUNT ER ON 'DATI'/
0        5     17 0 2116  042440
0        4      2 047117  023440
0        2      4 044  24 0 0047
00  70  0     02  043106  051105    EM6:    .ASCIZ  /BUFFER ADRS ER ON 'DATI'/
```

```
006276  040440  051104  020123
0[illegible]304  051105  047440  020116
[illegible]2  04[illegible]047  052101  023511
[illegible]     000
[illegible]     106  04[illegible]01  042514  EM7:   .ASCIZ  /FAILED TO INTR ON A 'DATO'/
[illegible]  0[illegible]104  04[illegible]  044440
[illegible]  [illegible]116  [illegible]2  047117
[illegible]  [illegible]  [illegible]0  040504
[illegible]  [illegible]  [illegible]47
[illegible]  [illegible]  [illegible]101  051525  EM10:  .ASCIZ  /STATUS ER ON 'DATO'/
[illegible]  [illegible]  [illegible]  042117
[illegible]  [illegible]  04[illegible]04  047524
[illegible]6  [illegible]47
[illegible]  0[illegible]27  042122  041440  EM11:  .ASCIZ  /WORD COUNT ER ON 'DATO'/
[illegible]6  0[illegible]517  052116  042440
[illegible]414  [illegible]2  047117  023440
[illegible]2  0[illegible]4  047524  000047
[illegible]430  0[illegible]2  043106  051105  EM12:  .ASCIZ  /BUFFER ADRS ER ON 'DATO'/
[illegible]436  0[illegible]0  051104  020123
0[illegible]444  051105  047440  020116
0[illegible]452  042047  052101  023517
0[illegible]460     000
0[illegible]461     104  052101  020101  EM13:  .ASCIZ  /DATA ER ON 'DATO'/
006466  051105  047440  020116
006474  042047  052101  023517
0[illegible]502     000
0[illegible]503     105  051122  041520  DH1:   .ASCIZ  /ERRPC   BUSADR  EXPCT   RCVD/
006510  020040  041040  051525
006516  042101  020122  042440
006524  050130  052103  020040
006532  051040  053103  000104
006540  051105  050122  020103  DH2:   .ASCIZ  /ERRPC   MEMADR  EXPCT   RCVD/
006546  020040  042515  040515
006554  051104  020040  054105
006562  041520  020124  020040
006570  041522  042126     000

        006576                          .EVEN
006576  001116  001122  001124  DT1:   $ERRPC,$BDADR,$GDDAT,$BDDAT,0
006604  001126  000000
                                ;;########################################################
                                ;'DBUF' IS THE WORKING AREA IN MEM FOR ALL NPR OPERATIONS
                                ;;########################################################
006610  000000                  DBUF:  0               ;1ST ADRS OF DATA BUFFER
        000001                         .END
```

```
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  = 0[illegible]00    411#    1296
BIT11  = 004000            410#
BIT12  = 010000            409#
BIT13  = 0[illegible]0     408#    1303
BIT14  = 0[illegible]0     407#
BIT15  = 1[illegible]0     406#
BIT2   = [illegible]4      429#
BIT3   = [illegible]10     428#
BIT4   = [illegible]0      427#
BIT5   = [illegible]0      426#
BIT6   = [illegible]00     425#
BIT7   = [illegible]       424#
[illegible] = [illegible]  423#
BIT9   = [illegible]0      422#
[illegible]TVEC= [illegible]14   436#
C[illegible] = 1[illegible]6     960     1291    1292    1311    1319    1378    1379    1570#
C[illegible]F [illegible]12      8[illegible]4   1034#
CR     = [illegible]15     346#    1123    1133
CRLF   = [illegible]0      347#    679     1094    1133
C[illegible]F [illegible]10      611     614     1035    1047    1644#
D[illegible]ISP = 17[illegible]70   353#    490     640
DH1    [illegible]3        521     527     533     539     544     550     556     562     568     574     1627#
DH2    [illegible]40       5[illegible]    1632#
D[illegible]PLA [illegible]1142  490#    640#    648#    1295#   1395#
[illegible] 0[illegible]0174     454#    648
[illegible]R 0[illegible]1320    [illegible]#   [illegible]2
[illegible]R 0013[illegible]6    [illegible]#   775#    811     815     888#    921     925
DRVCSR  001330             5[illegible]#   6[illegible]7#  705#    708     722#    723     738     741#    742     744     754#    755     776#
                           780     781#    783     788     792     793#    797     820#    825#    826     842#    843     859#
                           8[illegible]0   8[illegible]4   866#    868#    889     891#    895     896#    898     902     903#    907     950#
                           9[illegible]2#  9[illegible]3
DRVCT0  001334             601#    6[illegible]7   1015#   1023#
DRVCT2  001336             602#    661     1016#   1023    1024#
DRVC[illegible]R 001332    597#    655     704#    707     710     721#    753#    845     848#    862
[illegible]ECT 00[illegible]2    590#    6[illegible]8
[illegible]R 0[illegible]4       5[illegible]#   [illegible]1   774#    801     803     887#    911     913
D[illegible]R = 177570     352#    [illegible]9    639
DT1     00[illegible]576   522     5[illegible]8   534     540     545     551     557     563     569     575     581     1639#
EMTVEC= 0[illegible]0      441#    631#    632#
EM1     0[illegible]6      520     1575#
EM10    0[illegible]4      561     1610#
EM11    006400             567     1614#
```

DVDRA.P11 CROSS REFERENCE TABLE

```
EM12    006430            573    1618#
EM13    006451            579    1623#
EM2     006120            526    1581#
EM3     006161            532    1587#
EM4     006214            538    1592#
EM5     006240            543    1596#
EM6     006270            549    1601#
EM7     006321            555    1606#
EOPT    003464            962    1003#
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   = 104435          673    1568#
HT      = 000011          344#   1092    1133
ICOUNT  = 001346          609#    613#    963#    965#
IOTVEC= 000020            439#    613#    630#
LDLJF   = 003730          771    1046#
LF      = 000012          345#   1127    1133
MSTER   = 001344          608#    617#    619#    684     961
MTPS    = 106427          446#    690     768     777     881     892    1014    1025
MTST1   = 000012          701#   1006
MTST2   = 000134          737#
MTST3   = 000054          767#
PC      =X000007          345#    832#    959#    979#    982#    992#    997    1026#   1101#   1108#   1115#   1129#   1131#
                         1305#   1318#   1320#   1358#   1452#
PIRQ    = 177772          351#
PIRQVE= 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          346#    349
PSW     = 177776          349#
PWRVEC= 000024            440#
RDCHR   = 104437         1511    1571#
RDLIN   = 104410         1572#
RESVEC= 000010            435#
RMON    = 001356          613#    827     869     891     916     917     928     929
RSTVEC  = 000172          832     859    1033#
R0      =X000000          346#    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      =X000001          357#    652#    653     654#    658#    659     660#    702#    716#    929#    934#   1035#   1036#
                         1047#   1050#   1224    1237#   1238    1242    1266#
R2      =X000002          358#    930#    932     936#    939#    940#    945    1048#   1050    1054#   1057#   1058#   1225
                         1236#   1240#   1243    1250#   1251#   1252    1257#   1265#
```

```
R3      =X000003        359#   931*   937*  1049*  1055*  1166   1175*  1181*  1182*  1185*  1190*  1191*  1198
                       1201*  1226   1234*  1235*  1249*  1252*  1261*  1262*  1264*  1507   1508*  1509   1512*
                       1513   1517   1519   1521*  1523*
R4      =X000004        360#   1167   1169*  1170*  1171*  1172   1173*  1187   1189*  1197*  1200*
R5      =X000005        361#   769*   771*   882*   884*   1015   1017*  1034   1039*  1046   1053*  1168   1174*
                       1176*  1178*  1179*  1180*  1181   1199*  1227   1229*  1231*  1238*  1242*  1257   1263*
R6      =X000006        362#   364    623*   624*   625
R7      =X000007        363#   365
SAVE    001342          677#   686*   827*   829    845*   846    848    864*   865*   866    953*   954*   956
SETUP2  001552          623#   656
SETUP3  001616          659#   662
SETVEC  000002          769    [illegible]  1014#
SP      =X000006        [illegible]  [illegible]  637*   645*   649    681*   786    901    986*   1084*  1085   1096*  1099
                       1[illegible]  1[illegible]  1092   1094   1096   1102   1104*  1106*  1114*  1118   1122   1123   1127
                       1158*  1159   1160   1161*  1166*  1167*  1168*  1174   1199   1200   1201   1202*  1203*
                       1223*  1224*  1225*  1226*  1227*  1228*  1229   1232*  1245   1247*  1249   1259   1261
                       1263   1264   1265   1266   1267   1269*  1270*  1300   1330*  1335*  1336   1340*  1343*
                       1346   1349   1389   1393   1396*  1412*  1413*  1414   1421*  1424*  1425*  1429*  1430*
                       1434   1437*  1441   1443   1445   1446*  1453   1455   1457*  1458   1460*  1461*  1462*
                       1463*  1464*  1479*  1480*  1483*  1484*  1485   1489*  1490*  1491   1494   1496   1498*
                       1507*  1512   1523   1524*  1525*  1526*  1546*  1547
STACK = 001100          339#   627    681
START   001400          618    620#
START1  001364          457    617#
START2  001372          458    619#
STKLMT= 177774          350#
STST1   000572          686    841#
STST2   000736          858#
STST3   003014          880#
SWR     001140          489#   625    639*   641    647*   671    1296   1303   1308   1408   1445*
SWROK   000000         1305   1318#
SWREG   000176          455#   647    671    1408   1421
SW0     =000001         403#
SW00    =000001         393#   403
SW01    =000002         392#   402
SW02    =000004         391#   401
SW03    =000010         390#   400
SW04    =000020         389#   399
SW05    =000040         388#   398
SW06    =000100         387#   397
SW07    =000200         386#   396
SW08    =000400         385#   395
SW09    =001000         384#   394
SW1     =000002         402#
SW10    =002000         383#
SW11    =004000         382#
SW12    =010000         381#
SW13    =020000         380#
SW14    =040000         379#
SW15    =100000         378#
SW2     =000004         401#
SW3     =000010         400#
SW4     =000020         399#
SW5     =000040         398#
```

B04


```
SW6    = 000100       397#
SW7    = 0[illegible]0  396#
S[illegible]8  = 0[illegible]  395#
S[illegible]9  = 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= 0[illegible]014  437#
TYPCS  = 10[illegible]401  997   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#
XPR.M    001350       610#   774    775    776    806    807
$AUTOB   001134       486#   675*  1416   1537
$[illegible]ADR  001122  481#   707*   738*   783*   797*   803*   815*   898*   907*   913*   925*   944*  1639
$[illegible]DAT  001126  483#   710*   711    744*   745    780*   792*   795    801*   811*   812*   813    895*
                      902*   905    911*   921*   922*   923    943*  1639
$BELL'   001160       [illegible]#  1298  1314
$CHARC   004206      1099*  1109*  1116   1125*  1130#
$CKSWR   006236      1408#  1570
$CMTAG   001100       469#   622    623    631    635
$CM3   = 000000       499#
$CNTLG   [illegible]  1419   1532#
$CNTLU   [illegible]  1436   1531#
$CRLF    001165       501#  1098   1133   1306   1314   1329   1348   1353   1357   1447   1531
$DBLK    [illegible]  1234   1268   1276#
$DDAGN   [illegible]   981    990    996#
$DTBL    [illegible]  1237   1272#
$ENDAD   [illegible]   992#
$ENDCT   [illegible]   9[illegible]#
$ENDMG   003615       985   1000#
$ENULL   003612       988    999#
$EOP     003614       974#
$EOPCT   [illegible]   980#   984
$ERFLG   001102       472#  1293*  1314   1373   1394*  1398
$ERMAX   001115       478#  1398
$ERROR   [illegible]   631   12[illegible]0#
$ERRPC   001116       473#  1300*  1301*  1302   1314   1335   1639
$ERRTB   001170       518#  1343
$ER[illegible]TY  [illegible]010  1318  13[illegible]8#
$ERTTL   001112       476#  12[illegible]*  1314
$FILLC   001156       497#  1102   1133
$FILLS   001155       496#  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   003566       999#
$GTSWR   [illegible]  1420#  1568
$HD    = 000002       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    006222          1390    1395#
$PASS    001100          470#    977*    978*    986     999
$QUES    001164          500#    1133    1314    1466    1515    1531
$ROCHR   006520          1479#   1571
$RDDEC=  ****** U        1573
$POLIN   006640          1507#   1572
$[illegible]OCT= ****** U        1573
$[illegible]SZ = 000010          1500#
$[illegible]TNAD 003610          9[illegible]#
$[illegible]A  = ****** U        1573
$[illegible]ME= ****** 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#
$[illegible]AD 0[illegible]206           1387    1392#
$[illegible] = 124000    325     326#    327#    333     334     335     336     499     635     971     977     991     997
                         [illegible] 12[illegible]3 1284    1285    1286    1287    1296    1303    1308    1312    1314    1374    1380
                         1390    1392    1394    1398
$S[illegible]MK= 000000          1374
$T[illegible]B   [illegible]1146 492#    1401    1412    1429    1483    1489
$TKS     [illegible]1144 491#    1401    1410    1426    1450*   1481    1487
$TN    = [illegible]1    3[illegible]#   3[illegible]#
$TPB     [illegible]1152 4[illegible]#   11[illegible]*  1133
$T[illegible]FLG [illegible]1157 4[illegible]#   1[illegible]  1133
$T[illegible]S   [illegible]150  4[illegible]#   1[illegible]  1133
$[illegible]P    [illegible]12   633     1[illegible]#
$[illegible]   = [illegible]11   15[illegible]4#  1[illegible]#  1564#   1565#   1566#   1567#   1568    1569#   1570    1571#   1572#   1573#
$T[illegible]PAR [illegible]     1551    1[illegible]1#
$TSTAM   [illegible]2    471#    [illegible]*    1295    1314    1373    1392*   1395    1398
$TTYIN   [illegible]6    15[illegible]3  15[illegible]9  1526    1530#
$TYP[illegible]= ****** U        1[illegible]7
$TYPUS   [illegible]     1[illegible]#   1566
$TYPE    [illegible]2772 1[illegible]#   15[illegible]4  1562
$TY[illegible]C [illegible]4142  1101    1108    1115    1120#   1121    1452
$TYPEX   [illegible]10   1126    1128    1131#
$TYPOC   [illegible]     116[illegible]#  1563
$TYPON   [illegible]     11[illegible]2  1165#   1565
$TYPOS   [illegible]212  11[illegible]#  1564
$XTSTR   [illegible]19[illegible] 12[illegible]1#
$[illegible]ET4= [illegible]100  9[illegible]1#
$$FILL   004435          1159*   1163*   1173    1208#
```

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

```
C    EN       1#     446#
E   COM       1#     446#
E   OR     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
S    U        1#     446#     635#
T     P    1554#
TY   IN       1#     446#
TY    C       1#     446#     986
TYP   M       1#     446#     663
TY    M       1#     446#
TYPUCS        1#     315#     446#
TYPOCT        1#     446#    1335     1359     1421
TYPTXT        1#     446#
$ CHRE      462#
$   TN      462#
$   SCA       1#     446#
$    T        1#     446#
      T    1554#    1563     1564     1565     1566     1568     1570     1571     1572
     IP       1#     446#
.E   T        1#     315#     336
.H  JE        1#     315#
.KT11         1#
.    UP       1#     315#     616
.     I       1#     315#     329
.     O     336#
.    TI       1#
.    TB       1#
.    TH       1#
.   PTY       1#
.  STA        1#
.$ ATC        1#     315#     447
.    TA       1#     315#     462
.     D       1#
.     O       1#
.SDIV         1#
```

.SEOP 1# 315# 966
.SERRO 1# 315# 1277
.SERRT 1# 315# 1321
.SHLLT 1#
.SPOWE 1#
.SRNRO 1#
.SUDOC 1#
.SUDOX 1# 315#
.[illegible] 1# 315# 1398
.[illegible] 1#
.[illegible] 1#
.[illegible] 1#
.[illegible] 1#
.[illegible] 1# 315# 1368
.[illegible] 1#
.[illegible] 1#
.[illegible] 1# 315# 1538
.[illegible] 1#
.STYPO 1# 315# 1210
.STYPE 1# 315# 1063
.STYPO 1# 315# 1133
.SWOCA 1#
.1170 1#

G04

ADD	654	660	749	750	810	920	1090	1161	1171	1242	1343	1437	1446		
ASL	718	809	919	939	1057	1340	1341	1342	1460	1461	1462	1550			
ASLB	1247														
BCC	941	1059	1248												
BCS	720														
BEQ	685	709	712	724	743	746	752	756	789	796	802	814	828	844	847
	861	863	890	906	912	924	935	955	990	1093	1128	1188	1294	1297	1345
	1350	1363	1417	1444	1459										
BGT	981	1195	1256	1456	1497										
BIC	705	781	793	812	827	859	865	896	903	922	954	978	1185	1413	1430
	1457	1484	1490	1498											
BIS	1190	1191	1250	1251	1464										
BISB	1332														
BIT	708	723	742	751	755	788	843	860	889	1296	1303				
BITB	1116														
BLOS	1510														
BLT	1107	1196	1239	1255	1454	1495									
BMI	1246														
BNE	636	642	656	662	666	670	672	717	779	791	819	824	830	894	933
	938	949	957	962	964	1005	1038	1052	1056	1087	1095	1103	1117	1124	1186
	1244	1304	1333	1355	1409	1415	1435	1442	1449	1486	1492	1514	1520		
BPL	1081	1121	1184	1230	1260	1309	1411	1427	1482	1488					
BR	618	644	674	677	714	748	785	799	805	821	849	867	900	909	915
	927	942	1060	1083	1100	1110	1119	1126	1162	1177	1198	1241	1258	1338	1365
	1381	1387	1390	1438	1465	1467	1493	1516							
CLR	617	624	702	722	754	800	842	868	910	931	976	1024	1036	1049	1175
	1233	1276	1331	1424	1425										
CLRB	[illegible]	[illegible]	1099	1125	1262	1394	1521								
CMP	625	641	655	661	671	711	745	786	795	813	829	846	862	901	905
	923	932	956	1254	1388	1408	1414	1434	1441	1453	1455	1485	1491	1494	1496
	[illegible]														
CMPB	[illegible]	1074	1102	1123	1127	1416	1448	1513	1519						
COM	715	716	936	937	[illegible]	1055									
DEC	778	790	818	823	893	948	963	979	1004	1339					
DECB	1106	1109	1183	1194											
EMT	340														
HALT	453	1082	1310												
INC	665	719	934	940	977	1037	1051	1058	1189	1197	1240	1299	1463		
INCB	1129	1293	1392												
IOT	341														
JMP	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	997	1006							
JSR	769	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	992	1101	1108	1115	1305	1318	1452		
MOV	619	623	[illegible]	[illegible]	[illegible]	631	632	633	634	637	638	639	640	645	647
	648	649	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	659	681	683	687	703	704	706	707
	710	721	[illegible]	[illegible]	[illegible]	741	744	753	773	774	775	776	780	782	783
	787	792	794	[illegible]	[illegible]	803	806	807	811	815	817	822	826	845	848
	864	876	887	[illegible]	[illegible]	885	897	[illegible]	902	904	907	911	913	916	917
	921	925	[illegible]	[illegible]	[illegible]	943	944	945	947	953	965	982	986	989	1003
	1015	1016	1023	1034	[illegible]	1046	1047	1048	1050	1084	[illegible]	1089	1104	1158	1166
	1167	1168	1174	1181	1199	1200	1201	1202	1203	1223	1224	1225	1226	1227	1228
	1229	1234	1237	[illegible]	1263	1264	1265	1266	1267	1269	1270	1295	1300	1330	1335
	1344	1349	1354	[illegible]	[illegible]	[illegible]	1384	1386	1389	1393	1395	1396	1421	1445	1450
	1479	1480	1507	1508	1523	1524	1525	1526	1546	1547	1551				
MOVB	675	820	866	950	[illegible]	1114	1122	1159	1160	1163	1164	1165	1169	1172	1173

	1192	1232	1235	1249	1252	1261	1302	1412	1429	1483	1489	1512	1517	1549	
NFG	[illegible]	918	1170	1231											
[illegible]	[illegible]	975	993	994	995										
[illegible] SET	[illegible]	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	[illegible]	1301													
TRAP	1554	1563	1564	1565	1566	1568	1570	1571	1572						
TST	[illegible]	684	961	1088	1096	1118	1187	1243	1253	1308	1362	1385	1443	1458	1548
TSTB	[illegible]	1120	1245	1259	1410	1426	1481	1487							
.ASCII	[illegible]	501													
.ASCIZ	[illegible]	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													
.DS[illegible]L	1458														
.[illegible]	1	315	1401												
.E[illegible]	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	740	761	766	767	836	837	840	841	854	[illegible]	857	858	872
	873	879	[illegible]	[illegible]	970	971	973	976	981	984	985	[illegible]	991	997	999
	1000	1003	1010	1014	1020	1023	1029	1034	1042	1046	1066	[illegible]	1136	1213	1280
	1283	1293	1300	1305	[illegible]	1307	[illegible]	1313	1314	1315	1318	1324	1339	1368	1371
	1374	[illegible]	1391	[illegible]	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
	[illegible]	1569	1570	1571	1572	1573	1642	1644							
.EQUIV	340	341	349	[illegible]	365	394	395	396	397	398	399	400	401	402	403
	422	423	424	[illegible]	426	427	428	429	430	431					
.EVEN	679	[illegible]	1537	[illegible]											
.IF	316	[illegible]	[illegible]	[illegible]	404	432	456	464	468	470	499	[illegible]	[illegible]	616	622
	627	[illegible]	[illegible]	[illegible]	[illegible]	651	666	667	668	671	678	[illegible]	[illegible]	[illegible]	700
	727	728	[illegible]	[illegible]	[illegible]	760	765	766	835	836	[illegible]	[illegible]	[illegible]	854	856
	[illegible]	871	[illegible]	[illegible]	[illegible]	[illegible]	969	970	971	972	973	[illegible]	[illegible]	983	985
	[illegible]	991	[illegible]	[illegible]	[illegible]	[illegible]	1013	1019	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1065	1086
	[illegible]	1212	[illegible]	[illegible]	[illegible]	[illegible]	1303	1305	1306	[illegible]	[illegible]	[illegible]	[illegible]	1317	1323
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1379	[illegible]	1390	1392	1393	[illegible]	[illegible]	[illegible]	1400	1402
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1509	1513	1514	1530	[illegible]	[illegible]	1540	1546
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1567	[illegible]	1570	1571	1572	[illegible]	[illegible]	1643	
.IFF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	470	499	504	627	666	[illegible]	[illegible]	691	700	701
	728	[illegible]	[illegible]	[illegible]	[illegible]	761	766	767	836	837	840	[illegible]	[illegible]	855	857
	[illegible]	872	[illegible]	[illegible]	[illegible]	969	972	975	981	984	[illegible]	[illegible]	[illegible]	1020	1023
	[illegible]	1034	[illegible]	[illegible]	[illegible]	1136	1213	1280	1282	1296	[illegible]	[illegible]	[illegible]	1315	1318
	1324	1339	[illegible]	1371	[illegible]	[illegible]	1398	1401	1404	1472	[illegible]	[illegible]	[illegible]	1501	1510
	1514	1531	[illegible]	[illegible]	1642	1644									
.IFT	679	[illegible]	[illegible]	1474	1479										
.IFTF	679	[illegible]	[illegible]	1419	1472	1475									
.IIF	315	[illegible]	[illegible]	[illegible]	333	334	335	453	503	628	631	[illegible]	[illegible]	[illegible]	976
	977	[illegible]	[illegible]	[illegible]	1133	[illegible]	[illegible]	[illegible]	1286	1287	1291	1311	[illegible]	[illegible]	[illegible]
	1361	[illegible]	1378	[illegible]	1398	1401	1422	1523	1531	1537	1562	[illegible]	1564	[illegible]	[illegible]
	1568	1570	1571	1572											

.IRP 616 975 1223 1263 1292 1379
.LIST 1 315 446 453 499 616 635 667 668 679 976 991 1313 1374 1500
1554 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573
.MACRO 1 336 462 584 1554
.MCALL 315 446 635 668
.NLIST 1 315 446 453 499 616 635 667 668 679 976 991 1313 1374 1500
1554 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573
.PAGE 462 504 542 584 616 688 1007 1061
.PRINT 1
.RPT 453
.SBTTL 329 336 447 456 462 504 616 621 663 668 688 726 758 834 952
870 966 1007 1061 1063 1133 1210 1277 1321 1368 1398 1538 1554 1574
.TITLE 315
.WORD 453 454 455 460 470 473 474 475 476 479 480 481 482 483 484
485 488 489 490 980 983 998 1130 1209 1347 1352

ERRORS DETECTED: 0

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

J04

Spooler runtime 6 Seconds, 31 KCS, 177 disk reads, 3 disk writes, 47 pages