Department of Applied Analysis
and Computer Science

Technical Report CS 73 - 11

March 1973

DATA COMMUNICATIONS SUPPORT ROUTINES FOR THE NETWORKS LABORATORY

by

Carl Britney

This report in the University of Waterloo Computer Science Departmental Series is also available under separate cover, for the benefit of readers in the Computer Communications area who do not receive the Computer Science Series.

NAME DCI:

DESCRIPTION - DCI: is a non-terminal input device driver for the DC11 asynchronous interface. Binary data blocks are accepted by DCI: and sent to the system address. In detail DCI: when it is ready to accept a 256 word block sends a request to the sending end of the network. The other end will then begin sending the data byte by byte. The first byte is for indicating an end of file. The next 512 bytes are data. If the first byte indicates end, DCI will tell the system and a normal CLOSE will initiate. DCI: is the counterpart to DCO:

USES - together with DCO:, DCI: provides hands off file transfer between any two PDP-11 DOS systems. The transfer can take place in the same room with NULL MODEMS or across the country over a single telephone line. Also, other machines could support this type of transfer with a program to simulate these routines.

PROGRAMMING: DCI: is an input device with 4 units which map into the DC11 speeds. (See DCT: for a better explanation). Otherwise it can be used as any other DOS device.

ERRORS: no errors will evolve from DCI but the handler will loop until the line is connected.

NOTE DCO: and DCI: are being combined into one file DCF.

```
                        .TITLE DCI
        174000 RCSR=174000
        174002 RBUF=174002
        174004 TCSR=174004
        174006 TBUF=174006
        000000 R0=%0
        000001 R1=%1
        000006 SP=%6
        000005 R5=%5
        000007 PC=%7
                        ;
                        ;       DC11 INPUT HANDLER
                        ;
000000  000000  DCI:    .WORD 0             ;DVR VECTOR
000002  000334          .WORD 334           ;INPUT,ASCII,BIN
000004      020          .BYTE 20            ;256 WORD BUFFER
000005      166          .BYTE RXINT-DCI
000006      240          .BYTE 240
000007      015          .BYTE DC.OPN-DCI
000010      110          .BYTE DC.TRN-DCI
000011      072          .BYTE DC.CLS-DCI
000012  000000          .WORD 0
000014  114641  DC.DEV: .RAD50 /DCI/
                        ;
                        ;       OPEN ROUTINE
                        ;
000016  012737  DC.OPN: MOV #1,@#RCSR    ;SET READY
        000001
        174000
000024  012737          MOV #1,@#TCSR
        000001
        174004
000032  016700          MOV DCI,R0
        177742
000036  116001          MOVB 13(R0),R1  ;GET UNIT #
        000013
000042  042701          BIC #177774,R1  ;DROP GARBAGE
        177774
000046  006101          ROL R1
000050  006101          ROL R1
000052  006101          ROL R1
000054  050137          BIS R1,@#RCSR   ;CONVERT TO SPEEDS AND SET IN CSR
        174000
000060  050137          BIS R1,@#TCSR
        174004
000064  005726  DCOPN1: TST (SP)+
000066  000170          JMP @14(R0)     ;FINISHED OPEN OR CLOSE
        000014
                        ;
                        ;
                        ;       CLOSE ROUTINE CLEAR CSR
                        ;
000072  005037  DC.CLS: CLR @#RCSR
        174000
000076  005037          CLR @#TCSR
        174004
```

```
000102 016700          MOV DCI,R0
       177672
000106 000766          BR DCOPN1
                ;
                ;       TRANSFER START ROUTINE SET ADDRESSES AND WAIT FOR
                ;       CARRIER DETECT TO BEGIN RECEIVING
                ;
000110 016700 DC.TRN:  MOV DCI,R0
       177664
000114 016067          MOV 6(R0),WHERE ;SYSTEM BUFFER ADDRESS
       000006
       000160
000122 016001          MOV 10(R0),R1   ;GET WORD COUNT
       000010
000126 006301          ASL R1          ;NOW BYTE COUNT
000130 010167          MOV R1,WHEN     ;STORE
       000150
000134 032737          BIT #4,@#RCSR   ;CARRIER?
       000004
       174000
000142 001774          BEQ .-6         ;NO
000144 012737          MOV #6,@#TBUF
       000006
       174006
000152 105267          INCB SW
       000132
000156 052737          BIS #100,@#RCSR ;EN INTERRUPT
       000100
       174000
000164 000207          RTS PC          ;RETURN WAIT
                ;
                ;
000166 105767 RXINT:   TSTB SW         ;FIRST CHAR
       000116
000172 001410          BEQ RXINT1      ;NO  NORMAL
000174 105737          TSTB @#RBUF     ;END OF FILE CHAR
       174002
000200 100421          BMI RXINT2      ;YES FINISH
000202 105037          CLRB @#RBUF
       174002
000206 105067          CLRB SW
       000076
000212 000002          RTI
000214 005737 RXINT1:  TST @#RCSR      ;LOOK FOR ERROR
       174000
000220 100411          BMI RXINT2
000222 113777          MOVB @#RBUF,@WHERE
       174002
       000052
000230 005267          INC WHERE
       000046
000234 005267          INC WHEN
       000044
000240 002771          BGE RXINT2      ;END OF LINE
000242 000002          RTI
                ;
```

```
                        ;       ERROR OR END
                        ;
000244 042737 RXINT2:   BIC #100,@#RCSR ;DISABLE DEV
       000100
       174000
000252 013746           MOV @#44,-(SP)
       000044
000256 004536           JSR R5,@(SP)+
000260 016700           MOV DCI,R0
       177514
000264 016701           MOV WHEN,R1
       000314
000270 006201           ASR R1
000272 011160           MOV R1,16(R0)
       000016
000276 000170           JMP @14(R0)
       000014
                        ;
000302 000000 WHERE:    .WORD 0
000304 000000 WHEN:     .WORD 0
000306 000000 ERRORC:   .WORD 0
000310    000 SW:       .BYTE 0
       000001           .END
```

NAME - DCO:

DESCRIPTION - DCO: is a non-terminal output device driver for the DC11 interface unit. DCO: sends out 256 word data blocks at the request of the receiving end of the line. This is the counterpart or complement of DCI: In detail DCO: waits for an ACK character from the receiving end. Upon receipt of it a block of data is shipped out preceded by a null block indicating data. If DCO: is in an end of file state, the character 1ØØ will be sent instead. This will turn off the receiving end.

USES - see DCI:

PROGRAMMING - DCO: also has 4 speeds which are mapped into unit numbers Ø-3. Otherwise DCO: works as any other DOS device.

ERRORS - no errors will occur from DCO.

NOTE - DCO: and DCI: are being combined into one name DCF:

```
                        .TITLE DCO
        174?04 TCSR=174?04
        174?00 RCSR=174?00
        174?02 RBUF=174?02
        174?06 TBUF=174?06
        000?00 R0=%0
        000?01 R1=%1
        000?06 SP=%6
        000?05 R5=%5
        000?07 PC=%7
                        ;
                        ;       DC11 OUTPUT HANDLER
                        ;
?00000 ?00?00 DCO:      .WORD 0             ;DVR VECTOR
?00002 000332           .WORD 332           ;OUTPUT,ASCII,BIN,TERMINAL
000004     ?20          .BYTE 20                    ;256 WORD BUF
?00005     240          .BYTE TXINT-DCO
000006     247          .BYTE 240
?00007     ?16          .BYTE DC.OPN-DCO
000010     140          .BYTE DC.TRN-DCO
000011     376          .BYTE DC.CLS-DCO
000012 000000           .WORD 0
000014 ?14607 DC.DEV:   .RAD50 /DCO/
                        ;
                        ;       OPEN ROUTINE
                        ;
000016 ?12737 DC.OPN:   MOV #1,@#TCSR       ;SET REQUEST TO SEND
       000001
       174?04
000024 ?05?67           CLR SW
       000?70
000030 ?12737           MOV #1,@#RCSR               ;ENABLE RECEIVER
       ?00?01
       174?00
000036 ?16?70           MOV DCO,R0
       177?36
000042 ?16?01           MOVB 13(R0),R1      ;GET UNIT #
       000?13
000046 ?42?71           BIC #177774,R1      ;DROP GARBAGE
       177774
000052 ?06?01           ROL R1
000054 ?06?01           ROL R1
000056 ?06?01           ROL R1
000060 ?52?37           BIS R1,@#TCSR       ;CONVERT TO SPEEDS AND SET IN CSR
       174?04
000064 ?52?37           BIS R1,@#RCSR               ;ALSO REC
       174?00
000070 ?05?26 DCOPN1:   TST (SP)+
000072 ?00?70           JMP @14(R0)         ;FINISHED OPEN OR CLOSE
       ?00?14
                        ;
                        ;
                        ;       CLOSE ROUTINE CLEAR CSR
                        ;
000076 ?05?37 DC.CLS:   TSTB @#RCSR
       174?00
```

```
                BPL .-4
                MOVB @#RBUF,R5          ;GET THE CHAR
                CMPB R5,#6
                BNE DC.CLS              ;IF NOT ACK THEN IGNORE
                CLR R5
                MOVB STOP,@#TBUF
                CLR WHEN
                INC SW
                RTS PC
;
;       TRANSFER START ROUTINE SET ADDRESSES AND WAIT FOR
;       CLEAR TO SEND TO BEGIN XMITTING
;
DC.TRN: MOV DCO,R0
                MOV 6(R0),WHERE ;SYSTEM BUFFER ADDRESS
                MOV 10(R0),R1   ;GET WORD COUNT
                ASL R1          ;NOW BYTE COUNT
                MOV R1,WHEN     ;STORE
                DEC WHEN
                BIT #2,@#TCSR   ;CLEAR TO SEND ?
                BEQ .-6         ;NO
DCTRN1: TSTB @#RCSR             ;OTHER END READY ?
                BPL .-4                 ;NO   WAIT
                MOVB @#RBUF,R5
                CMPB R5,#6
                BNE DCTRN1              ;UNLESS 6 IGNORE
                CLR R5
                BIS #100,@#TCSR ;EN INTERRUPT
                MOVB START,@#TBUF       ;SEND START CODE FIRST
                RTS PC                  ;WAIT FOR DONE
;
;
;
TXINT:  INC WHEN
```

```
000244 002376          BGE TXEND
000246 117737          MOVB @WHERE,@#TBUF
       000340
       177566
000254 105267          INC WHERE
       000232
000260 000002          RTI
000262 005767 TXEND:   TST SW                  ;CLOSE OR TRAN
       000232
000266 001402          BEQ TXEND1              ;TRAN
000270 105237          CLR @#TCSR
       177564
000274 013746 TXEND1:  MOV @#44,-(SP)
       000044
000300 004536          JSR R5,@(SP)+
000302 116700          MOV DCO,R0
       177772
000306 000170          JMP @14(R0)
       000014
                       ;
000312 000000 WHERE:   .WORD 0
000314 000000 WHEN:    .WORD 0
000316     000 START:  .BYTE 0
000317     200 STOP:   .BYTE 200
000320 000000 SW:      .WORD 0
       000001          .END
```

NAME DCT:

DESCRIPTION: DCT is an ASCII terminal handler for the DC11 asynchronous interface on the PDP-11. Both terminal output and echo input are supported by DCT: All speeds supported by the DC11 are available without any special control functions.

On OUTPUT - Character lines of ASCII are put onto the terminal device until either the end of file or a Control X is typed on the keyboard. To continue from a Control X simply type another one. Tabs are converted to spaces, Rubout is ignored as are all illegal control characters.

On INPUT - The character '>' is typed on the device to indicate that it is ready to receive one line. Each input character is checked and echoed if valid. Tabs are converted to spaces. A control U will restart the current line. A rubout deletes the previous character in the line. Any valid terminator, LF, VT or FF will disable the input until the system has received the current line. Then the whole sequence recommences. DCT: has its own internal line buffer for input. All operations are done on it until a terminator initiates the transfer to the system.

USES - DCT: can replace the teletype for both input and output applications. There are many terminal devices that can operate at speeds up to 1800 BAUD. DCT: can operate these devices, whereas the KB: driver cannot.

- The DC11 can operate through any standard modem device. With DCT: one could remotely access the PDP-11 from any distance through telephone lines.

- DCT: could become the console handler for the PDP-11 DOS system. With this the user could access DOS from his home or another computer could initiate jobs on the remote PDP-11.

PROGRAMMING - DCT is a DOS compatable driver for terminal devices. It can be employed where KB: is used with one exception: DCT requires a unit number Ø to 3. This unit number maps into the speed setting for the DC11.

Eg. if one has a DC11 AA wired for speeds
Ø - 110 BAUD
1 - 134.5 "
2 - 150 "
3 - 300 "

DCT: or DCTØ: will drive the DC11 at 110 BAUD
DCT1: DCTØ: will drive the DC11 at 134.5 "
DCT2: DCTØ: will drive the DC11 at 150 "
DCT3: DCTØ: will drive the DC11 at 300 "

DCT: expects the DC11 to start at address 174ØØØ and interrupt vector at 3ØØ. Any other configuration requires alteration of DCT.

- Other than the speed setting, DCT can be used as any other system I/O device.

ERRORS - no errors occur directly from DCT: .

- A listing of DCT follows: -

```
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
S202=0
;
;       DC11 TERMINAL DEVICE DRIVER
;
        .TITLE DCT
        .GLOBL DCT
;
;       DESIGNED TO DRIVE ANY ASCII TERMINAL DEVICE
;
;       FROM DOS 11 , ALSO COULD BE USED FOR A
;
;       CONSOLE DEVICE UNDER DOS
;
;       ---DEVICE VECTOR---
;
DCT:    .WORD 0             ;BUSY IND. AND DDB POINTER
        .WORD 427           ;TERM,IN,OUT,ASCII,MULTI USER
        .BYTE 6             ;120 OCTAL BYTES BUFFER
        .BYTE TRNXX-DCT     ;RCVR INT ADDRESS DISP
        .BYTE 240           ;PRIORITY = 5
        .BYTE 0
        .BYTE DCTRN-DCT     ;TRAN ROUTINE
        .BYTE 0,0,0
DCTDEV: .RAD50 /DCT/        ;NAME OF DEVICE
;
TRNXX:  JMP TRANR           ;INT ROUTINE ENTRY
;
;       --- TRAN ROUTINE ---
;
;       1-      SET SPEED , DATA ADDRESSES ,I/O SWITCH
;
DCTRN:  MOV DCT,R0          ;GET DDB ADDR
        MOVB 13(R0),R1      ;GET UNIT #
        BIC #177774,R1
        ASL R1
        ASL R1
        ASL R1              ;CONVERT TO SPEED FOR DC11
        INC R1              ;ADD ENABLE BIT
        MOV R1,DCENAB       ;KEEP FOR LATER
        MOV PC,R1
        ADD #TRANX-.,R1
        MOV R1,@#304        ;SET TX INT VECTOR
        MOV #240,@#306
        MOV PC,R1
        ADD #BUFR-.,R1
        MOV R1,PBUFR        ;GENERATE ADDRESSES FOR LATER USE
        ADD #117,R1
        MOV R1,PBUFRE
```

```
        MOV PC,R1
        ADD #BUFCR-.,R1
        MOV R1,PBUFCR
;
;       SET SYSTEM BUFFER AND COUNT
;
        MOV 6(R0),WHERE ;ADDRESS
        MOV 10(R0),R1
        ASL R1
        MOV R1,WHEN     ;BYTE COUNT
        CLRB SWITCH
        BIT #4,12(R0)   ;IN OR OUT
        BEQ .+6         ;OUT SW=0
        INCB SWITCH
;
;       INIT VARIABLES AND SWITCHES
;
INITIT: MOV R0,-(SP)
        MOV PBUFR,R0
        CLRB (R0)+
        CMP R0,PBUFRE
        BLT .-6
        MOV (SP)+,R0
        MOV DCT,R0
        CLRB ECHOSW
        CLRB WAITSW
        CLRB STOPSW
        CLRB TABSW
        CLRB TABCNT
        CLRB DONESW
        MOV PBUFR,START         ;INIT INPUT BUFFER PTRS
        MOV START,ECHO
        DEC ECHO
;
;       START DC 11
;
        MOV DCENAB,@#174000
        MOV DCENAB,@#174004
;
;       WAIT FOR CARRIER BEFORE GOING ON
;
INITT1: MOV @#174000,TEMP0
        BIT #4,TEMP0
        BEQ INITT1
        BIS #100,@#174000
;
;       CHECK DIRECTION SWITCH AND BRANCH IF OUTPUT
;
        TSTB SWITCH
        BEQ TRNX00      ;OUTPUT BRANCH
;
;       INIT INPUT SEQUENCES
;
        INCB ECHOSW
        MOVB #'←,@#174006
        BIS #100,@#174004
        RTS PC
```

```
;
;       --- END OF TRAN SETUP ---
;
;
;       INIT OUTPUT  BY FAKING INTERRUPT AND DROPPING TO INT TRN
;
TRNX00: MOV (SP),-(SP)
        MOV @#177776,2(SP)
;
;       TRANSMITTER INTERRUPT ROUTINE
;
;       FIRST DETERMINE DIRECTION
;
TRANX:  BIC #100,@#174004       ;DROP INT ON ENTRY
        TSTB TABSW              ;TAB IN PROG.?
        BEQ TRANX0              ; NO ! GO ON
;
;       OUTPUT ANOTHER SPACE UNTIL TABCNT GIVES OUT
;
TRANX4: MOVB #40,@#174006
        INCB TABCNT             ;ONE LESS
        BPL TRANX3              ;DONE !
TRANX9: BIS #100,@#174004       ;EN INT.
        RTI
TRANX3: CLRB TABSW              ;RESET WHEN DONE
        INC WHEN
        BR .-20
TRANX5: INCB TABSW              ;ENTRY FOR INIT TABS
        BISB #370,TABCNT
        BR TRANX4               ;SEND FIRST SPACE
;
;
;       NEXT LOOP POSITION - LOOK FOR DIR.
;
TRANX0: TSTB SWITCH             ;IN OR OUT ?
        BNE TRANE               ;IN ! GO TO ROUTINE
;
;       OUTPUT ONLY SECTION OF TX INTERRUPT RTN
;
        TSTB STOPSW             ;DO WE STOP OUTPUTTING ?
        BEQ TRANX8              ;NO !!
;
;       STOP CURRENT LINE BY FORCING CR LF
;
        MOV PBUFCR,WHERE
        MOV #-2,WHEN
        CLRB STOPSW
;
;       CHECK FOR END OF LINE, GET CHAR , OUTPUT SAME
;
;       UNLESS TAB OR TERMINATOR INWHICH CASE WAIT
;
;       IF USER HAS TYPED ^X.
;
TRANX8: TST WHEN                ;DONE ?
        BPL DCEND               ;YES !!!
;
```

```
;
TRANX1: MOVB @WHERE,TCHAR
        INC WHERE                       ;UP OND NOW !
        BICB #200,TCHAR                 ;GET CHAR AND DROP PARITY
        CMPB TCHAR,#177                 ;RUB OUT ?
        BEQ TRANX7                      ;YES IGNORE !
        CMPB TCHAR,#40                  ;VALID ?
        BLT TRANX2                      ;MAYBE NOT
;
;       SEND OUT CHARACTER
;
TRANX6: INCB TABCNT
        INC WHEN
        MOVB TCHAR,@#174006             ;SEND IT
        JMP TRANX9
;
;       CHECK FOR TAB ,CR , AND TERMINATOR
;
TRANX2: CMPB TCHAR,#11                  ;TAB ?
        BEQ TRANX5                      ;YES GO INIT TAB
        BLT TRANX7                      ;IGNORE IF <
        CMPB TCHAR,#15                  ;CR ?
        BEQ TRANX6                      ;YES SEND IT
        BGT TRANX7                      ;IGNORE IF >
;
;       TERMINATOR FOUND  WAIT IF USER HAS INDICATED
;
        MOV @#-2,-(SP)
        CLR @#-2                        ;DROP PRIORITY FOR POS ↑C
        TSTB WAITSW                     ;WAIT UNTIL WAITSW = 0
        BNE .-4
        MOV (SP)+,@#-2                  ;RESTORE PRIORITY
        CLRB TABCNT
        CMPB TCHAR,#12  ;LINE FEED ?
        BNE TRANX7                      ;DO NOT WRITE IF NOT
        BR TRANX6+4
;
;       IGNORE CHAR
;
TRANX7: INC WHEN
        BR TRANX8                       ;GO GET ANOTHER CHAR
;
;       END OF LINE REACHED RETURN TO SYSTEM
;
DCEND:  MOV @#44,-(SP)
        JSR R5,@(SP)+
        MOV DCT,R0
        JMP @14(R0)                     ;END
;
;       ECHO PORTION OF TRANSMIT INTERRUPT ROUTINE
;
;
TRANE:  INC ECHO                        ;BUMP ECHO POINTER
        CMP ECHO,START                  ;UP TO END YET ?
        BLT TRANE0                      ;NO !  CONTINUE
;
;       CHECK TO SEE IF A COMPLETE LINE YET
```

```
;
;       IF NOT - JUST RETURN
;
        CLRB ECHOSW             ;ECHO DISABLE
        TSTB DONESW             ;COMPLETE LINE
        BNE TRANFL              ;YES GO FILL BUFFER
        TSTB STOPSW             ;↑U MATBE ???
        BNE TRANE3              ;YES - RESET LINE
        RTI                     ;OTHERWISE RETURN
;
;
;       ECHO CHARACTER AFTER CHECKING FOR SPECIALS
;
;       SUCH AS RUBOUT,CR,TAB,LF ETC.
;
TRANE0: MOVB @ECHO,TCHAR        ;GET CHAR
        BICB #200,TCHAR
        CMPB TCHAR,#177         ;RUBOUT ???
        BEQ RUBOUT              ;YES !! SPECIAL PROCESSING
        CMPB TCHAR,#40          ;VALID CHAR
        BLT .+20                ;IGNORE ECHO IF NOT!
TRANE1: MOVB TCHAR,@#174006     ;ECHO CHAR
        INCB TABCNT             ;AND  SET TAB UP
        JMP TRANX9
;
;       CONTROL CHAR CHECK FOR ↑U,ETC
;
;
        CMPB TCHAR,#25          ;↑U ???
        BEQ TRANE2              ;YES !  GO AWAY
        CMPB TCHAR,#11          ;TAB ?
        BEQ TRANX5              ;YES INIT TAB SPACING
        BLT TRANE               ;IGNORE IF LESS
        CMPB TCHAR,#15          ;C R ??
        BGT TRANE               ;IGNORE IF GREATER
        BNE TRANE1              ;TERM CHAR - OK.
        MOVB #12,@START         ;END OF LINE
        INC START               ;C.R. ADD LINE FEED
        BR TRANE1               ;NOW ECHO C.R.
;
;       CONTROL U -  RESET LINE AND ECHO '↑U' ETC
;
TRANE2: INCB STOPSW             ;TELL US ABOUT IT
        MOV PBUFCR,START        ;SET TO ↑U PRINT
        MOV START,ECHO
        INC START
        DEC ECHO
        DEC ECHO
        BR TRANE0       ;START IT
;
;       AFTER ↑U DONE COME HERE
;
TRANE3: CLRB STOPSW             ;RESET SWITCH
        MOV 2(SP),@#177776      ;SET FOR RTS  NOT RTI
        MOV (SP)+,(SP)
        JMP INITIT              ;RESET
;
```

```
;       RUBOUT ROUTINE  BACK UP ONE DELETED CHARACTER
;
RUBOUT: MOV R0,TEMP0            ;KEEP R0
        MOV ECHO,R0             ;WHERE WERE WE
        DEC ECHO
        DEC ECHO                ;THIS IS WHERE WE END
        CMP ECHO,PBUFR          ;UNLESS BACKED TOO FAR
        BLT RUB2                ;WE DID !
;
;       MOV CHARS UNTIL END REACHED
;
RUB0:   CMP R0,START            ;END /
        BGE RUB1                ;YES STOP
        MOVB (R0)+,-2(R0)       ;GRAB NEXT CHAR
        BR RUB0
;
;       END  CLEAN UP POINTERS
;
RUB1:   DEC R0
        DEC R0          ;NOW POINTS TO NEW END
        MOV R0,START
        MOV TEMP0,R0            ;RESTORE R0
        MOV #'/,@#174006        ;OUTPUT *
        JMP TRANX9
;
;       JUST RESET AND RETURN
;
RUB2:   MOV PBUFR,ECHO
        MOV ECHO,START
        CLRB ECHOSW
        RTI
;
;       SEND A LINE TO THE SYSTEM
;
TRANFL: MOV @#44,-(SP)
        JSR R5,@(SP)+   ;STORE REGS
        MOV WHERE,R1
        MOV WHEN,R2
        MOV PBUFR,R3
FIL1:   MOVB (R3)+,(R1)+        ;MOVE CHAR
        CMP R3,PBUFRE           ;END ?
        BLT FIL1
;
;       RETURN TO SYSTEM
;
FIL2:   MOV OCT,R0
        JMP @14(R0)
;
;       RECEIVER INTERRUPT ROUTINES
;
;       KEEP DATA AND CHECK DIRECTION
;
TRANR:  MOV @#174000,TEMP0      ;GET CSR
        BMI DCERR               ;IF ERROR  GO THERE
        MOVB @#174002,RCHAR     ;GET DATA
        BICB #200,RCHAR         ;MAKE POSITIVE
        TSTB SWITCH             ;IN OR OUT
```

```
        BNE TRANR1                      ;IN  JUMP -
;
;       OUTPUT TEXT  TRAP ↑U AND ↑X ONLY
;
;       SET STOPSW IF ↑U
;
;       SET WAITSW IF ↑X
;
        CMPB RCHAR,#25                  ; STOP LINE IN PROG ?
        BNE TRANR0+2                    ;NO TRY THE OTHER
        INCB STOPSW
TRANR0: RTI                 ;RETURN
        CMPB RCHAR,#30
        BNE TRANR0                      ;NOT THE OTHER
        INCB WAITSW                     ;INCREMENT SWITCH
        BNE .+4
        RTI
        MOVB #-1,WAITSW
        RTI
;
;
;       RECEIVING TEXT ROUTINES
;
TRANR1: TSTB WAITSW                     ;ARE WE STOPPED ?
        BNE TRANR0                      ;YES  DO NOTHING
        CMPB RCHAR,#40                  ;VALID ?
        BGE TRANR2                      ;YES  STORE IT
        CMPB RCHAR,#25
        BEQ TRANR3
        CMPB RCHAR,#15
        BGT TRANR0                      ;IGNORE
        CMPB RCHAR,#11
        BEQ TRANR2                      ;STORE IF EQUAL
        BLT TRANR4
;
;       TERMINATOR CHARACTER FOUND
;
        INCB DONESW
TRANR3: INCB STOPSW
        INCB WAITSW
;
;       STORE CHAR
;
TRANR2: CMPB RCHAR,#141                 ;UPPERCASE CHECK
        BLT TRANR5
        CMPB RCHAR,#172
        BGT TRANR5
        TSTB UCASE
        BEQ TRANR5
        BICB #40,RCHAR
TRANR5: MOVB RCHAR,@START
        INC START
        TSTB ECHOSW                     ;ARE WE ECHOING
        BNE TRANR0                      ;YES  JUST EXIT
        INCB ECHOSW                     ;OTHERWISE INITIATE
        JMP TRANE0
TRANR4: CMPB RCHAR,#1                   ;CONTROL A?
```

```
        BNE .+6                 ;NO !
        INCB UCASE              ;SET UPPER CASE ONLY SWITCH
        CMPB RCHAR,#2           ;CONTROL B ?
        BNE TRANRØ
        CLRB UCASE
        BR TRANRØ
;
;       ERROR STOP
;
DCERR:  MOV TEMPØ,-(SP)
        MOV #S2Ø2,-(SP)
        IOT
;
;       DATA BUFFER
;
BUFR:   .=.+122
PBUFR:  .WORD Ø          ;POINTER TO BUFFER
PBUFRE: .WORD Ø          ;POINTER TO ITS END
        .ASCII /↑U/
BUFCR:  .BYTE 15,12
PBUFCR: .WORD Ø
;
;       SWITCHES
;
STOPSW: .BYTE Ø
WAITSW: .BYTE Ø
SWITCH: .BYTE Ø
ECHOSW: .BYTE Ø
TABSW:  .BYTE Ø
TABCNT: .BYTE Ø
DONESW: .BYTE Ø
TCHAR:  .BYTE Ø
RCHAR:  .BYTE Ø
UCASE:  .BYTE Ø
        .EVEN
;
;       DATA AREAS
;
TEMPØ:  .WORD Ø
DCENAB: .WORD Ø
START:  .WORD Ø
ECHO:   .WORD Ø
WHERE:  .WORD Ø
WHEN:   .WORD Ø
;
;
;
        .END
```

NAME DV:

DESCRIPTION - DV: is a device driver for the DIVA 2314 Disk Unit and Controller. Handling up to 4 drives (UNITS Ø-3) DV: reads and writes data blocks of 512 PDP-11 words onto the diskpack. DV: works under the DOS I/O system and has valid blocks from Ø to 56677 octal or $(24ØØØ)_{10}$ blocks of 512 words for its storage per unit. At the present DV: is a non-system device but in the near future DV: will be used as a system disk with its own DOS system on unit Ø.

USES - DV: provides DOS with a huge random storage area of 24ØØØ blocks of 512 words. Under the file system as many as 125 users may be enabled on a single unit as opposed to 14 users on a DF: system or 62 users on a DK: system.

PROGRAMMING - DV: is another DOS device after running a zeroing program. Any valid I/O operation is valid except the zero switch in PIP. However, due to the 512 word buffers required for transfers, 12K of core should be the minimum configuration. For initialization two steps are necessary. The first step is to format the diskpack to 512 word blocks using the DIVA supplied format routine.

WORDSIZE	= 1000	Octal
#PHYSICAL SECTORS/LOGICAL SECTORS	= 4	"
#LOGICAL SECTORS/TRACK	= 6	"
#CYL	= 313	"
#HEADS	= 24	"

Step 2 is to run the program DVFMAT.LDA.

DVFMAT.LDA - this program initializes a file system on the disk pack on unit Ø. MFD blocks #1 & #2 are written. All bitmaps are then written out. The file system uses the first 37 octal blocks. This program should only be used once per pack as PIP can now take care of file deletions etc.

After step 2 the pack is ready for use. Note that [1,1] and [200,200] do not exist after formatting and initialization.

ERRORS - only 3 possible errors can be generated directly from DV:

FØ35 xxxxxx - illegal block number. The block #xxxxxx is too big for the disk.

F044 xxxxxx - disk interface error. xxxxxx is the CSR of the interface.

F045 xxxxxx - device not ready i.e. wrong unit #.

- A listing of DV: follows: -

```
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
;
;       DISK DRIVER FOR DIVA DISK UNIT
;
;       VERSION ONE
;
;       FEATURES        SINGLE USER
;                       512 WORD BLOCKS / DISK
;                       4 UNITS HANDLED
;
;
        .TITLE DV
        .GLOBL DV
;
;       DEVICE ASSIGNMENTS
;
DCSR=164000
DDBR=164002
DMAR=164004
DWCR=164006
DCBR=164010
DSSR=164012
DAIR=164014
DUSR=164016
;
;       CBR COMMAND BITS
;
DIR=0
READ=40000
WRITE=100000
SPEC=140000
MOD0=0
MOD1=10000
MOD2=20000
MOD3=30000
;
;       CONTROL FUNCTIONS
;
INCHAD=1
RECAL=2
RESETH=20
SEEK=40
CLRHDA=100
;
;       DEVICE VECTOR
;
DV:     .WORD 0             ;DDB POINTER
        .WORD 102037        ;FACILITIES
        .BYTE 40            ;512 WORD BLOCKS
        .BYTE DV.INT-DV     ;IN ROUTINE
```

```
        .BYTE 240       ;PR = 5
        .BYTE 0
        .BYTE DV.TRN-DV ;TRA ROUTINE
        .BYTE 0
        .WORD 0
DV.DEV: .RAD50 /DV/
        .WORD 1         ;FIRST MFD BLOCK POINTER
        .WORD 0,0,0,0   ;4 DRIVES/DEVICE
;
;       INTERRUPT HANDLER FOR DV:
;
DV.INT: MOV @#DCSR,TEMP0        ;KEEP STATUS REG
        BMI DVERR               ;IF NEG  THEN ERROR
        TSTB TEMP0              ;R/W
        BPL DVSEEK              ;NO !
        BIS #200,@#DCSR ;GET RID OF DONE BIT
        BIC #100,@#DCSR         ;DROP INT
        MOV @#44,-(SP)
        JSR R5,@(SP)+
        MOV DV,R0
        JMP @14(R0)             ;RETURN DONE
;
;       CHECK SEEK OPERATION
;
DVSEEK: MOV #DIR+MOD3+CLRHDA,@#DCBR     ;RESET ATTN
        BIT #4000,@#DSSR        ;DRIVE READY ?
        BNE DVERR
        MOV PTRCFM,@#DMAR       ;SET C BUF IN MAR
        MOV BCOUNT,@#DWCR
        MOV COMND0,@#DCBR       ;LOAD HEAD NOW!
        MOV COMND1,@#DCBR       ;READ OR WRITE CONFIRM
        INC @#DCSR              ;GO !!!
        RTI
;
;       CALL ERROR ROUTINE WITH F44
;
DVERR:  MOV TEMP0,-(SP)         ;SHOW CSR
        MOV #1444,-(SP)         ;F 44
        IOT
;
;       TRANSFER ROUTINE
;
;       TWO STEPS       1-SEEK POSITION
;                       2-INITIATE R/W
;
DV.TRN: MOV #DIR+MOD3+CLRHDA+RESETH,@#DCBR      ;DROP POS INT
        MOV DV,R0       ;GET DDB
DECODE: MOV 4(R0),R1            ;GET BLOCK #
        CMP R1,#56677   ;CHECK SIZE
        BLE DCODE0      ;O.K.
;
;       ILLEGAL BLOCK # FOR DEVICE
;
        MOV R1,-(SP)
        MOV #1435,-(SP)
        IOT
;
```

```
;       DECODE CYLINDER,TRACK AND SEGMENT ADDRESSES
;
;
DCODE0: MOV CNT1,R3             ;FIRST DIVISOR
        MOV DIV1,R2             ;
        CLR R4
DCODE1: CMP R2,R1               ;DOES IT DIVIDE ?
        BGT DCODE2              ;NO
        SUB R2,R1
        INC R4
DCODE2: CLC
        INC R3
        BPL DCODE6              ;DONE FIRST #
        ROR R2
        ROL R4
        BR DCODE1
;
;       STORE SEGMENT
;
DCODE6: ASL R1
        MOV R1,SEGMT
;
;       START NEXT
;
        MOV R4,R1
        MOV CNT2,R3             ;SECOND DIVISOR
        MOV DIV2,R2             ;
        CLR R4
DCODE3: CMP R2,R1               ;DOES IT DIVIDE ?
        BGT DCODE4              ;NO
        SUB R2,R1
        INC R4
DCODE4: CLC
        INC R3
        BPL DCODE7
        ROR R2
        ROL R4
        BR DCODE3
DCODE7: MOV R1,TRACK
        MOV R4,CYLNDR
        JMP SEEKIT
SEGMT:  .WORD 0
TRACK:  .WORD 0
CYLNDR: .WORD 0
DIV1:   .WORD 30000
CNT1:   .WORD 177765
DIV2:   .WORD 50000
CNT2:   .WORD 177765
TEMP0:  .WORD 0
BCOUNT: .WORD 0
UNIT:   .WORD 0
COMND0: .WORD 0
COMND1: .WORD 0
PTRCFM: .WORD 0
CFMBUF: .WORD 0,0,0,0
;
;       SET UP SEEK AND CONFIRM BUFFER
```

```
;
;       ALSO SET UP ALL R/W COMMANDS NOW !
;
SEEKIT: MOV @#DSSR,TEMP0          ;GET HDWR STATUS
        BIC #143777,TEMP0         ;DROP SOME OF IT
        BNE DVERR                 ;ERROR IF NOT READY
;
;       SET UNIT # AND SELECT
;
        MOVB 13(R0),R1
        BIC #177770,R1
        MOV R1,R3
        CLR R2
        INC R2
SEEK1:  DEC R1
        BLE .+6
        ASL R2
        BR SEEK1
        MOV R2,UNIT               ;UNIT SET
        MOV R3,R2
        ADD #SPEC+MOD0,R2
        MOV R2,@#DCBR             ;SELECT UNIT
        BIT #2000,@#DSSR          ; SELECT OK ?
        BNE SEEK2                 ;YES
        MOV DV.DEV,-(SP)
        MOV #1445,-(SP)
        IOT
;
;       RESET HEAD
;       SET CYLINDER AND HEAD REGISTERS
;
SEEK2:  MOV CYLNDR,R1
        MOV TRACK,R2
        ADD #DIR,R2
        ADD #DIR+MOD2,R1
        MOV R1,@#DCBR
        MOV R2,COMND0             ;SET TRACK AND CYLINDER
;
;       BUILD UP CONFIRM BUFFER FOR I/O
;
        MOV PC,R1
        ADD #CFMBUF-.,R1
        MOV R1,PTRCFM             ;GENERATE ABS ADDRESS
;
;       CHECK DIRECTION AND GENERATE READ OR WRITE
;
        MOV #READ,R1              ;START WITH READ
        BIT #4,12(R0)             ;IN OR OUT
        BNE .+6                   ;IN !  O.K.
        MOV #WRITE,R1
        ADD #MOD3,R1              ;+CONFIRM
        ADD SEGMT,R1              ;+PHYSICAL SEG.
        MOV R1,COMND1             ;STORE INSTRUCTION FOR R/W
;
;       NOW FINISH COMFIRM BUFFER
;
        MOV PTRCFM,R2             ;GET POINTER
```

```
MOV TRACK,R1                ;GET TRACK PT
SWAB R1                     ;PUT IN UPPER BYTE
ADD CYLNDR,R1               ;ADD IN CYL PTR
MOV R1,(R2)+                ;FIRST ENTRY
MOV 10(R0),R3               ;GET WORD COUNT
ASL R3
MOV R3,BCOUNT               ;SAVE FOR READ / WRITE
MOV #177000,R1              ;GET WORD COUNT
ASL R1                      ;CONVERT TO WORD COUNT FIRST
ASL R1
ASL R1
ASL R1
MOV SEGMT,R3     ;DIVIDE SEGMT BY 2 AND ADD TO R1
CLC
ASR R3
ADD R3,R1
MOV R1,(R2)+                ;SECOND WORD
CLR R1
SUB -4(R2),R1
SUB -2(R2),R1
MOV R1,(R2)+                ;THIRD WORD IS CHECKSUM
MOV 6(R0),(R2)              ;LAST WORD IS DATA BUFFER
MOV UNIT,@#DAIR             ;ENABLE UNIT IN AIR
BIS #100,@#DCSR             ;AND INTERRUPT
MOV #DIR+MOD3+SEEK+RESETH,@#DCBR        ;START SEEK
RTS PC                      ;GO WAIT FOR DONE
.END
```

Name DV: Version 2

DV: - Device driver and file system for 2314 type disk systems, version 2.

DESCRIPTION - DV: provides the PDP-11 with large scale random access storage. The disk park is divided into 2 logical units each with 24000 blocks of 256 words. The even unit (i.e. 0,2,4, or 6) is physically the first 100 cylinders. The odd unit becomes the last 100 cylinders. The file system designed for this unit will permit full DOS access for 125 users per logical unit. DV: will permit up to 4 drives without modification.

USES - A large scale storage area for PDP-11s. One disk will hold the equivalent of 12 RK05 disks at a cost of only 4 RK packs.

PROGRAMMING - DV: is the logical name of the device and may be used in the same manner as any other DOS file storage device except for initialization. DV: must first have a formatted disk before any transfers take place. The necessary procedure follows:

1) Format the disk pack to 400_8 words per logical sector, 2_8 physical sectors per logical, 14_8 logical sectors per track.

2) Run the program DVFMAT.LDA. This program writes all Master File blocks and the bit maps.

   In particular after formatting the disk will have the following following blocks written on <u>both</u> logical units.

   - BLOCK Ø reserved for possible bootstrap

   - BLOCK 1 Master File Directory #1

   - BLOCK 56645 Master File Directory #2

   - BLOCK 56646 Master File Directory #3

   - BLOCKS 56647 - 56677 BIT MAPS 1-25

From this point on, PIP may be used to enable users etc., as any other DOS device.

NOTE 1 - DV: must be added to DOS with a new system generation using the DEC supplied program MODS.

NOTE 2 - DV: does not yet support a DOS system stand alone. The ideal system device for use with DV: would be the RF11 dixed head disk.

ERRORS - Two possible errors stem from DV:

F035 - illegal block number
F044 - disk interface error
F045 - device not ready

These errors were made fatal because the DOS system recovery methods are not applicable to this device.

- A listing of DV: and DUFMAT follows -

```
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
;
;       DISK DRIVER FOR DIVA DISK UNIT
;
;       VERSION TWO
;
;       FEATURES        SINGLE USER
;                       256 WORD BLOCKS / DISK
;                       8 UNITS HANDLED
;
;
        .TITLE DV
        .GLOBL DV
;
;       DEVICE ASSIGNMENTS
;
DCSR=164000
DDBR=164002
DMAR=164004
DWCR=164006
DCBR=164010
DSSR=164012
DAIR=164014
DUSR=164016
;
;       CBR COMMAND BITS
;
DIR=0
READ=40000
WRITE=100000
SPEC=140000
MOD0=0
MOD1=10000
MOD2=20000
MOD3=30000
;
;       CONTROL FUNCTIONS
;
INCHAD=1
RECAL=2
RESETH=20
SEEK=40
CLRHDA=100
;
;       DEVICE VECTOR
;
DV:     .WORD 0             ;DDB POINTER
        .WORD 102037        ;FACILITIES
        .BYTE 20            ;256 WORD BLOCKS
```

```
        .BYTE DV.INT-DV ;IN ROUTINE
        .BYTE 240       ;PR = 5
        .BYTE 0
        .BYTE DV.TRN-DV ;TRA ROUTINE
        .BYTE 0
        .WORD 0
DV.DEV: .RAD50 /DV/
        .WORD 1         ;FIRST MFD BLOCK POINTER
        .WORD 0,0,0,0   ;4 DRIVES/DEVICE
        .WORD 0,0,0,0   ;PLUS 4 MORE FOR VERSION 2
;
;       INTERRUPT HANDLER FOR DV:
;
DV.INT: MOV @#DCSR,TEMP0        ;KEEP STATUS REG
        BMI DVERR               ;IF NEG  THEN ERROR
        TSTB TEMP0              ;R/W
        BPL DVSEEK              ;NO !
        BIS #200,@#DCSR ;GET RID OF DONE BIT
        BIC #100,@#DCSR         ;DROP INT
        MOV @#44,-(SP)
        JSR R5,@(SP)+
        MOV DV,R0
        JMP @14(R0)             ;RETURN DONE
;
;       CHECK SEEK OPERATION
;
DVSEEK: MOV #DIR+MOD3+CLRHDA,@#DCBR     ;RESET ATTN
        BIT #4000,@#DSSR        ;DRIVE READY ?
        BNE DVERR
        MOV PTRCFM,@#DMAR       ;SET C BUF IN MAR
        MOV BCOUNT,@#DWCR
        MOV COMND0,@#DCBR       ;LOAD HEAD NOW!
        MOV COMND1,@#DCBR       ;READ OR WRITE CONFIRM
        INC @#DCSR              ;GO !!!
        RTI
;
;       CALL ERROR ROUTINE WITH F44
;
DVERR:  MOV TEMP0,-(SP)         ;SHOW CSR
        MOV #1444,-(SP)         ;F 44
        IOT
;
;       TRANSFER ROUTINE
;
;       TWO STEPS       1-SEEK POSITION
;                       2-INITIATE R/W
;
DV.TRN: MOV #DIR+MOD3+CLRHDA+RESETH,@#DCBR      ;DROP POS INT
        MOV DV,R0       ;GET DDB
DECODE: MOV 4(R0),R1            ;GET BLOCK #
        CMP R1,#56677   ;CHECK SIZE
        BLE DCODE0      ;O.K.
;
;       ILLEGAL BLOCK # FOR DEVICE
;
        MOV R1,-(SP)
        MOV #1435,-(SP)
```

```
        IOT
;
;       DECODE CYLINDER,TRACK AND SEGMENT ADDRESSES
;
;
DCODEØ: MOV CNT1,R3             ;FIRST DIVISOR
        MOV DIV1,R2             ;
        CLR R4
DCODE1: CMP R2,R1               ;DOES IT DIVIDE ?
        BGT DCODE2              ;NO
        SUB R2,R1
        INC R4
DCODE2: CLC
        INC R3
        BPL DCODE6              ;DONE FIRST #
        ROR R2
        ROL R4
        BR DCODE1
;
;       STORE SEGMENT
;
DCODE6: ASL R1
        MOV R1,SEGMT
;
;       START NEXT
;
        MOV R4,R1
        MOV CNT2,R3             ;SECOND DIVISOR
        MOV DIV2,R2             ;
        CLR R4
DCODE3: CMP R2,R1               ;DOES IT DIVIDE ?
        BGT DCODE4              ;NO
        SUB R2,R1
        INC R4
DCODE4: CLC
        INC R3
        BPL DCODE7
        ROR R2
        ROL R4
        BR DCODE3
DCODE7: MOV R1,TRACK
        MOV R4,CYLNDR
        JMP SEEKIT
SEGMT:  .WORD Ø
TRACK:  .WORD Ø
CYLNDR: .WORD Ø
DIV1:   .WORD 3ØØØØ
CNT1:   .WORD 177765
DIV2:   .WORD 5ØØØØ
CNT2:   .WORD 177765
TEMPØ:  .WORD Ø
BCOUNT: .WORD Ø
UNIT:   .WORD Ø
COMNDØ: .WORD Ø
COMND1: .WORD Ø
PTRCFM: .WORD Ø
CFMBUF: .WORD Ø,Ø,Ø,Ø
```

```
;
;       SET UP SEEK AND CONFIRM BUFFER
;
;       ALSO SET UP ALL R/W COMMANDS NOW !
;
SEEKIT: MOV @#DSSR,TEMPØ       ;GET HDWR STATUS
        BIC #143777,TEMPØ      ;DROP SOME OF IT
        BNE DVERR              ;ERROR IF NOT READY
;
;       SET UNIT # AND SELECT
;
        MOVB 13(RØ),R1
        CLC             ;CLEAR CARRY IF ANY
        ROR R1          ;GET EVEN UNIT ONLY
        BCC .+1Ø        ;NO CARRY  = EVEN UNIT
        ADD #144,CYLNDR ;ODD UNIT DISPLACE UNIT
        BIC #177774,R1          ;DROP GARB.
        MOV R1,R3
        CLR R2
        INC R2
SEEK1:  DEC R1
        BLE .+6
        ASL R2
        BR SEEK1
        MOV R2,UNIT             ;UNIT SET
        MOV R3,R2
        ADD #SPEC+MODØ,R2
        MOV R2,@#DCBR           ;SELECT UNIT
        BIT #2ØØØ,@#DSSR        ; SELECT OK ?
        BNE SEEK2               ;YES
        MOV DV.DEV,-(SP)
        MOV #1445,-(SP)
        IOT
;
;       RESET HEAD
;       SET CYLINDER AND HEAD REGISTERS
;
SEEK2:  MOV CYLNDR,R1
        MOV TRACK,R2
        ADD #DIR,R2
        ADD #DIR+MOD2,R1
        MOV R1,@#DCBR
        MOV R2,COMNDØ           ;SET TRACK AND CYLINDER
;
;       BUILD UP CONFIRM BUFFER FOR I/O
;
        MOV PC,R1
        ADD #CFMBUF-.,R1
        MOV R1,PTRCFM           ;GENERATE ABS ADDRESS
;
;       CHECK DIRECTION AND GENERATE READ OR WRITE
;
        MOV #READ,R1            ;START WITH READ
        BIT #4,12(RØ)           ;IN OR OUT
        BNE .+6                 ;IN !  O.K.
        MOV #WRITE,R1
        ADD #MOD3,R1            ;+CONFIRM
```

```
        ADD SEGMT,R1            ;+PHYSICAL SEG.
        MOV R1,COMND1           ;STORE INSTRUCTION FOR R/W
;
;       NOW FINISH COMFIRM BUFFER
;
        MOV PTRCFM,R2           ;GET POINTER
        MOV TRACK,R1            ;GET TRACK PT
        SWAB R1                 ;PUT IN UPPER BYTE
        ADD CYLNDR,R1           ;ADD IN CYL PTR
        MOV R1,(R2)+            ;FIRST ENTRY
        MOV 10(R0),R3           ;GET WORD COUNT
        ASL R3
        MOV R3,BCOUNT           ;SAVE FOR READ / WRITE
        MOV #177400,R1          ;GET WORD COUNT
        ASL R1                  ;CONVERT TO WORD COUNT FIRST
        ASL R1
        ASL R1
        ASL R1
        MOV SEGMT,R3    ;DIVIDE SEGMT BY 2 AND ADD TO R1
        CLC
        ASR R3
        ADD R3,R1
        MOV R1,(R2)+            ;SECOND WORD
        CLR R1
        SUB -4(R2),R1
        SUB -2(R2),R1
        MOV R1,(R2)+            ;THIRD WORD IS CHECKSUM
        MOV 6(R0),(R2)          ;LAST WORD IS DATA BUFFER
        MOV UNIT,@#DAIR         ;ENABLE UNIT IN AIR
        BIS #100,@#DCSR         ;AND INTERRUPT
        MOV #DIR+MOD3+SEEK+RESETH,@#DCBR        ;START SEEK
        RTS PC                  ;GO WAIT FOR DONE
        .END
```

```
R0=%0
SP=%6
PC=%7
        .TITLE DVFMAT
;
;       INIT DIVA DISK FOR PDP 11 FILE SYSTEM
;
;
START:  MOV #LINK,-(SP)
        EMT 6                   ;INIT DV:
;
;       WRITE OUT MFD #1 AT BLOCK 1
;
        MOV #TRAN,R0
        MOV #1,(R0)+
        MOV #MFD1,(R0)
        TST -(R0)
        JSR PC,TRA
;
;       WRITE OUT MFD #2
;
        MOV #56645,(R0)+
        MOV #MFD2,(R0)
        TST -(R0)
        JSR PC,TRA
;
;
;       WRITE OUT MFD #3
;
        MOV #56646,(R0)
        CLR MFD2                ;END OF CHAIN
        JSR PC,TRA
        MOV #MAP,TRAN+2
;
;       NOW BIT MAPS ONE AT A TIME
;
LOOP:   INC MAP
        INC MAP+2
        INC TRAN                ;BL 56647 FIRST
        CMP TRAN,#56677         ;LAST ONE
        BEQ LOOP2               ;YES
        JSR PC,TRA
        CLR MAP+10              ;DROP BLOCK 0,1
        BR LOOP
;
;       WRITE OUT LAST MAP
;
LOOP2:  MOV #177777,MAP+174
        MOV #177777,MAP+176
        CLR MAP                 ;LAST IN CHAIN
        JSR PC,TRA
        EMT 60
;
;
TRA:    MOV #TRAN,-(SP)
        MOV #LINK,-(SP)
        EMT 10
```

```
        MOV #LINK,-(SP)
        EMT 1
        RTS PC
;
;
LINK:   .WORD 0,0,1
        .RAD50 /DV/
;
;
TRAN:   .WORD 0,0,400,2,0
;
;
MFD1:   .WORD 56645
        .WORD 5,56647
        .WORD 56647,56650,56651,56652,56653,56654,56655,56656
        .WORD 56657,56660,56661,56662,56663,56664,56665,56666
        .WORD 56667,56670,56671,56672,56673,56674,56675,56676
        .WORD 56677
        .=MFD1+2004
;
;
MFD2:   .WORD 56646
        .=MFD2+2004
;
;
MAP:    .WORD 56647,0,74,56647,3
        .=MAP+2004
        .END START
```

NAME MTS:

DESCRIPTION: MTS: is a device driver for the AMPEX 9 track tape unit and custom interface connected to our 11/20. MTS: handles all I/O operations and file system functions for the tape unit. Since magnetic tape is sequential with no central directory, the file system is different to that of any other DEC device. The file system is comprised of a sequence of files followed by 2 tape marks. Each file has as its first record a 7 word label similar to that of the file block used in all file operations. All successive records are 256 words long. The last record of the file is followed by a tape mark. So tape marks separate each file and 2 tape marks end the tape.

An OPEN command searches the tape labels for a match. Each request for data transfers a complete block of 256 words. CLOSE will write the 2 tape marks if a new file were created.

USES - MTS: permits DOS file operations onto magnetic tape. Thus, the tape unit becomes another bulk storage for DOS. Because of its nature, only one file is allowed open at any time. MTS: is another DOS device and may be used in the same way as DT:,DK:, DF:, etc.

PROGRAMMING - MTS: operates files outside of the regular DOS file system. This means that PIP commands such as /DI, /DE will not work. To overcome this deficiency 2 small programs are available to assist.

MTFMAT.LDA - reformats the tape with 2 tape marks, essentially cleaning the tape.

MTDIR.LDA - reads the tape labels and lists the directories onto a logical device DEV: which must be assigned prior to running MTDIR:

Otherwise MTS: can be programmed as any other single user file - oriented device.

ERRORS - 2 errors will occur in MTS:

FØ12 - means that either a file exists when it shouldn't or that no file exists for input.

F032 xxxxxx - indicates a hardware error on the tape. xxxxxx is the CSR status of the device. All errors are normally unrecoverable.

- A listing of MTS follows: -

```
R0=%0
R1=%1
R2=%2
SP=%6
PC=%7
R5=%5
MTCSR=177540
MTDBR=177542
MTMAR=177544
MTBCR=177546
MTBLK=177550
;
;                INPUT OUTPUT HANDLER FOR AMPEX UNIT
;
        .TITLE MTS
        .GLOBL MTS
MTS:    .WORD 0                 ;DVR VECTOR START
        .WORD 336               ;OPEN CLOSE, ASCII BIN, INPUT OUTPUT
        .BYTE 20                ;256 WORD BUFFER
        .BYTE MT.INT-MTS        ;START OF INT ROUTINE
        .BYTE 300               ;PR =6
        .BYTE MT.OPN-MTS        ;OPEN OFFSET
        .BYTE MT.TRN-MTS        ;TRANSFER OFFSET
        .BYTE MT.CLS-MTS        ;CLOSE OFFSET
        .WORD 0                 ;NO SPEC FUNCTIONS
MT.DEV: .RAD50 /MTS/            ;NAME OF DEVICE
;
;       OPEN ROUTINE INITIATE
;
;       GET FILE NAME EXT ETC. SEARCH TAPE FOR SUCH OR END OF TAPE
;       WHICHEVER COMES FIRST
;
;       CALL APPROPRIATE ERRORS IF NECESSARY
;
MT.OPN: MOV MTS,R0              ;GET ADDRESS OF DDB VECTOR
        MOV @#177776,PRSAVE     ;KEEP CALLING STATUS FOR ERROR
        MOV #400,BLOCK          ;RESET BLOCK COUNTER
        MOV #1,OPEN             ;SIGNAL OPEN
        CLR CLOSE               ;MAKE SURE OF NO CLOSE
        MOV PC,R1               ;GENERATE ADDRESS OF LABEL AREA
        ADD #FILE2-.,R1
        MOV R1,PFILE2           ;STORE IT FOR LATER USE
        MOV 4(R0),R1            ;GET POINTER TO FILBLK
        MOV R1,PFILE1           ;STORE FOR LATER USE
        MOVB -2(R1),OPENCD      ;KEEP OPEN CODE
;
;       CHECK FILE NAME FOR UIC AND PROTECT CODES
;
OPNA:   TST 6(R1)               ;CHECK UIC FOR ZEROS
        BNE TAG1                        ;NOT DEFAULT
        MOV @#440,6(R1)         ;REPLACE ZERO WITH CURRENT UIC
TAG1:   TST 10(R1)              ;CHECK PROTECT CODE
        BNE     TAG2
        MOV #233,10(R1)         ;REPLACE ZERO WITH 233
TAG2:   CLR OPNCDE              ;SET FIRST JMP CODE
        CLR @#MTBLK             ;ZERO BLOCK ADDRESS
```

```
        MOV #102,@#MTCSR        ;REWIND TAPE FUNCTION
        RTS PC                  ;RETURN UNTIL ACTION
;
;       CLOSE INITIATE ROUTINE
;
;       IF OPENI THEN STRAIGHT RETURN
;
;       ELSE WRITE 2 TAPE MARKS AND FINISH
;
MT.CLS: CMP OPENCD,#2           ;OPENO ?
        BEQ TAG3                        ;YES CONTINUE
        TST (SP)+               ;CLEAR CALL FROM STACK
        MOV MTS,R0              ;GET DDB ADDRESS
        JMP @14(R0)             ;RETURN DONE
TAG3:   MOV #1,CLOSE            ;SIGNAL CLOSE
        CLR CLSCDE              ;SET JUMP CODE
        MOV #105,@#MTCSR        ;WRITE TAPE MARK
        RTS PC                  ;WAIT
;
;       INTERRUPT ROUTINE START  CHECK FOR OPEN ERROR CLOSE
;       IN THAT ORDER THEN JUMP TO APPROPRIATE ROUTINE
;
MT.INT: BIT #2000,@#MTCSR               ;BUSYEADY ???
        BNE .-6                         ;YES WAIT
        CMP #300,@#MTCSR                ;LOOK FOR READY  NOT POSITIONED
        BNE TAG40               ;GO ON IF NOT
        INCB BLOCK              ;SET ONE BLOCK ON
        BIS #400,BLOCK          ;KEEP BIT 8 UP
        MOV BLOCK,@#MTBLK               ;GIVE IT TO REG
        MOV #106,@#MTCSR                ;SEARCH TO IT
        RTI
TAG40:  TST OPEN                ;IF OPEN GO TO SPECIAL PART
        BEQ TAG4
        JMP MTIOPN
TAG4:   TST @#MTCSR             ;LOOK FOR ERROR
        BPL TAG5                        ;NO ERROR
        JMP HERROR
TAG5:   TST CLOSE               ;SAME AS OPEN
        BEQ TAG6
        JMP MTICLS
;
;       INCREMENT BLOCK COUNTER AND TAKE SYSTEM EXIT
;
TAG6:   JMP OPN3                ;EXIT FINISHED
;
;
;       TRANSFER ROUTINE LOAD VARIABLES AND POINTERS
;       AND INITIATE BLOCK TRANSFER.
;
MT.TRN: MOV MTS,R0              ;GET DDB ADDRESS
        MOV 6(R0),@#MTMAR       ;GET DATA ADDRESS INTO MAR
        MOV 10(R0),R1           ;GET WORD COUNT
        COM R1                  ;MAKE POSITIVE
        ADD #1,R1               ;INCREASE BY ONE WORD
        ASL R1                  ;MULTIPLY BY TWO
        MOV R1,@#MTBCR          ;STORE BYTE COUNT IN DEV
        MOV 12(R0),R1           ;GET FUNCTION BITS
```

```
        ROR R1
        ROR R1
        ROR R1
        BMI MT.OUT              ;OUTPUT BIT ON
        MOV #101,@#MTCSR        ;READ DATA
        RTS PC                  ;WAIT FOR ACTION
MT.OUT: MOV @6(R0),@#MTDBR      ;SET FOR WRITE BY MOVING UP ONR
        ADD #2,@#MTMAR          ;WORD IN MEMORY
        MOV #104,@#MTCSR
        RTS PC                  ;WRITE AND WAIT
;       ********** OPEN SERVICE ROUTINES **********
;
;
;       JUMP TO OPEN ROUTINES
;
MTIOPN: TST @#MTCSR             ;CHECK FOR ERROR
        BPL MTIPN1              ;NO ERROR
        CMP OPNCDE,#4           ;WHAT WERE WE DOING --
        BEQ TAG7                        ;WE WERE READING A LABEL
        JMP HERROR              ; ERROR IN STATUS
TAG7:   CMPB @#MTDBR,#23                ;END OF TAPE ???
        BEQ TAG8
        JMP HERROR                      ;STILL ERROR
TAG8:   CLR FIND                ;SET INDICATOR
        JMP     OPN6            ;GO TO FINISH ROUTINE
;
;       NOW EXECUTE JUMP
;
MTIPN1: MOV R0,SAVE             ;KEEP R0
        MOV PC,R0
        ADD #OPNJMP-.,R0        ;GENERATE JUMP ADDRESS
        ADD OPNCDE,R0
        MOV R0,JMP
        MOV SAVE,R0
        JMP @JMP
OPNJMP: JMP OPN0
        JMP OPN1
        JMP OPN2
        JMP OPN3
JMP:    .WORD 0
;
;       TAPE REWOUND  READ FIRST LABEL
;
OPN0:   MOV #4,OPNCDE           ;NEXT JUMP
        MOV #16,@#MTBCR         ;7 WORD LABEL
        MOV PFILE2,@#MTMAR              ;LABEL BUFFER ADDRESS
        MOV #101,@#MTCSR                ;READ LABEL
        RTI                             ;RETURN TO WAIT
;
;       CHECK LABEL I MATCH JUMP OUT ELSE INITIATE
;       SEARCH TO TAPE MARK TO GET NEXT LABEL
;
OPN1:   CLR OPNCDE              ;JUMP CODE
        MOV R0,SAVE
        MOV R1,SAVE+2
        MOV PFILE1,R0
        MOV PFILE2,R1
```

```
        CMP (R0)+,(R1)+ ;CHECK LABEL
        BNE OPN11
        CMP (R0)+,(R1)+
        BNE OPN11
        CMP (R0)+,(R1)+
        BNE OPN11
        CMP (R0)+,(R1)+
        BNE OPN11
        JMP ERROR2              ;CHECK PROTECT CODE NOW
;
;       LABEL NOT A MATCH RESUME SEARCHING
;
OPN11:  CLR @#MTBLK
        MOV #400,BLOCK          ;RESET BLOCK COUNT
        MOV #107,@#MTCSR        ;SEARCH TO TAPE MARK
        MOV SAVE,R0
        MOV SAVE+2,R1
        RTI
;
; FOUND LABEL OR EDN OF TAPE
;
OPNFND: MOV #1,FIND             ;SIGNAL MATCH
        MOV SAVE,R0
        MOV SAVE+2,R1
;
;
OPN6:   TST FIND
        BEQ OPN0                ; L END OF TAPE BRANCH
;
;       MUST BE OPENI OR ELSE ERROR
;
        CMP OPENCD,#4           ;OPENI?
        BEQ TAG9
        JMP ERROR1              ;ERROR IF NOT
TAG9:   JMP OPN3                ;FINISHED OPENI
;
;       CHECK FOR OPEN0 IF NOT - ERROR
;
OPN0:   CMP OPENCD,#2           ;OPEN0 ?
        BEQ TAG10
        JMP ERROR1              ;ERROR IF NOT
;
;       BACKSPACE ONE BLOCK BEFORE WRITING NEW LABEL
;
TAG10:  MOV #10,OPNCDE          ;JMP  POINTER
        MOV #400,BLOCK          ;RESET BLOCK COUNT
        CLR @#MTBLK             ;BACK UP ONE BLOCK
        MOV #106,@#MTCSR                ;SEARCK BACK ONE BLOCK
        RTI
;
;
; WRITE NEW LABEL
;
OPN2:   MOV #14,OPNCDE          ;LAST JUMP POINTER
        MOV #16,@#MTBCR         ;16 BYTES OUT
        MOV @PFILE1,@#MTDBR     ;SEND FIRST WORD OF LABEL
        MOV PFILE1,@#MTMAR
```

```
        ADD #2,@#MTMAR  ;GET CORRECT ADDRESS
        MOV #104,@#MTCSR        ;WRITE
        RTI
;
;       END  OF OPENO
;
OPN3:   CLR OPEN                ;DISABLE OPEN
        MOV @#44,-(SP)          ;REG SAVE ROUTINE
        JSR R5,@(SP)+           ;CALL IT
OPN20:  MOV MTS,R0              ;DDB USED
        JMP @14(R0)             ;TAKE RETURN IN DDB VECTOR
;
;     ****************CLOSE SERVICE ROUTINES ***************
;
;       JMP TO ROUTINES
;
MTICLS: MOV R0,SAVE             ;REQUIRE R0
        MOV PC,R0               ;MAKE UP JMP ADDRESS
        ADD #CLSJMP-.,R0
        ADD CLSCDE,R0
        MOV R0,JMP
        MOV SAVE,R0
        JMP @JMP
CLSJMP: JMP CLS0
        JMP CLS1
;
;       WRITE SECOND TAPE MARK ON TAPE
;
CLS0:   MOV #4,CLSCDE
        MOV #105,@#MTCSR                ;WRITE TAPE MARK
        RTI
;
;       FINISHED CLOSING ON OPENO FILE
;
CLS1:   CLR CLOSE
        JMP OPN3                ;END CALL
;
;       DATA FOR HANDLER
;
SAVE:   .WORD 0,0               ;T STORE FOR REGS.
FILE2:  .WORD 0,0,0,0
        .WORD 0,0,0,0           ;TAPE FILE NAME
PFILE1: .WORD 0                 ;POINTER TO FILBLK
PFILE2: .WORD 0                 ;POINTER TO FILE2
FIND:   .WORD 0
BLOCK:  .WORD 0                 ;BLOCK COUNTER
OPNCDE: .WORD 0
OPENCD: .WORD 0
OPEN:   .WORD 0
CLSCDE: .WORD 0
CLOSE:  .WORD 0
PRSAVE: .WORD 0          ;PRIORITY SAVE
ERRORC: .BYTE 0
PRCT2:  .BYTE 0
;
;       ERROR ROUTINES FOR HANDLER
;
```

```
;       INVALID OPEN  IE END OF TAPE WHEN OPENI
;                     OR FILE FOUND ON OPENO
;
;       USER PROTECT CODE INVALID FOR READ
;
;       HARDWARE ERROR IN TAPE UNIT
;
;       END OF FILE ENCOUNTERED ON READ
;
;         IF THE ERROR JUMP IN USERS FILBLK IS NOT ZERO
;       THEN CONTROL IS PASSED THERE ON OPEN ERRORS .
;
;       AN END OF FILE BHT IS SET IN THE LINKBLK WHEN
;       ENCOUNTERED .
;
;       ---INVALID OPEN---
;
ERROR1: MOVB #2,ERRORC          ;ERROR CODE 2
ERROR4: DEC PFILE1                      ;POINT TO ERROR CODE BYTE
        MOVB ERRORC,@PFILE1             ;GIVE IT ERROR CODE
        DEC PFILE1                      ;NOW POINTS TO ERROR RET
        DEC PFILE1
        DEC PFILE1
        TST @PFILE1             ;IF ZERO FATAL ERROR
        BNE ERRJMP
        ADD #4,PFILE1           ;RESET PFILE1
        MOV PFILE1,-(SP)        ;GIVE ADDRESS
        MOV #1412,-(SP)         ;FATAL 12 ERROR
        IOT
;
;       JUMP TO USER DESTINATION IN CORE
;
ERRJMP: CLR @MTS                ;FREE DDB
        MOV PRSAVE,@#177776     ;USER PRIORITY
        CMP (SP)+,(SP)+         ;CLEAR INTERRUPT FROM STACK
        MOV @PFILE1,PFILE1      ;POINT TO USER ROUTINE
        JMP @PFILE1             ;EXIT TO USER ROUTINE
;
;       ---PROTECT CODE ERROR ROUTINE---
;
;       IF  OPENO IN PROGRESS IGNORE SINCE ERROR ANYWAY
;
;       IF UIC IN CORE AND UIC IN R1 MATCH IGNORE ALSO
;
;       OTHERWISE CHECK FOR VIOLATION IF NONE GO TO
;       OPNFND
;       ELSE CALL ERROR ROUTINE ABOVE WITH CODE 6
;
ERROR2: CMPB #2,OPENCD          ; OPENO
        BNE .+6
        JMP OPNFND              ;YES EXIT
        CMP -(R1),@#440         ;DO UICS MATCH
        BNE .+6
        JMP OPNFND              ;YES EXIT
;
;       CHECK UIC DIFFERENCES
;
```

```
        MOVB 2(R1),PRCT2                ;GET CODE
        CMPB 1(R1),@#44Ø                ;DO GROUPS MATCH
        BNE ERRNSA              ;NO !
        ASRB PRCT2
        ASRB PRCT2
        ASRB PRCT2              ;SHIFT GROUP CODE TO RIGHT
ERRNSA: BICB #37Ø,PRCT2
        CMPB PRCT2,#3           ;CHECK IT
        BGT     .+6
        JMP OPNFND              ;PROTECT CODE OK
        MOVB #6,ERRORC
        JMP ERROR4              ;ERROR 6 IN FILBLK
;
;       HARDWARE ERROR ENCOUNTERED
;
;       LOOK FOR END OF FILE MARK
;
HERROR: CMPB #23,@#MTDBR                ;END OF FILE ?
        BNE HERR1                       ;NO--REAL ERROR
HERR2:  MOV @#44,-(SP)                  ;SET TO END
        JSR R5,@(SP)+
        MOV MTS,RØ                      ;GET DDB ADDR
        MOV 1Ø(RØ),16(RØ)               ;NO DATA TO R/W
        JMP OPN2Ø                       ;EXIT
;
;       UNRECOVERABLE ERROR FATAL 32
;
HERR1:  MOV @#MTCSR,-(SP)
        MOV #1432,-(SP)
        IOT
        .END
```

```
        .TITLE MTDIR
;
;       LIST DIRECTORY OF MAG TAPE ONTO DEV: ASSIGNED BEFORE RUNNING
;
MTCSR=177540
MTDBR=177542
MTMAR=177544
MTBCR=177546
MTBLK=177550
SP=%6
        .GLOBL CVT
;
;
START:  MOV #LINK,-(SP)         ;INIT DEV:
        EMT 6
        CLR @#MTBLK             ;BLOCK ADDRESS Ø ON DRIVE
        MOV #FILE,-(SP)         ;OPEN DEV:
        MOV #LINK,-(SP)
        EMT 16
        MOV #LINK,-(SP)
        EMT 1
        MOV #2,@#MTCSR          ;REWIND TAPE TO LOAD POINT
        TSTB @#MTCSR
        BPL .-4
        TST @#MTCSR
        BMI ERROR
;
;
;       READ A LABEL OR END OF TAPE
;
;       CONVERT IT TO COMPLETE ASCII CHARACTERS
;       TO BE OUTPUT ONTO
;       WHATEVER DEVICE THE USER HAS DEFINED
;
START1: BIT #2000,@#MTCSR               ;WAIT UNTIL NOT BUSY
        BNE .-6
        MOV #14,@#MTBCR         ;READ LABEL
        MOV #BUFFER,@#MTMAR
        MOV #1,@#MTCSR
        TSTB @#MTCSR
        BPL .-4
        TST @#MTCSR             ;EXIT ON ANY ERROR
        BMI ERROR
        MOV #7,@#MTCSR          ;SEARCH TO NEXT TAPE MARK
        MOV @#BUFFER,-(SP)              ;CONVERT FILENAME EXT TO ASCII
        MOV #BUFF,-(SP)
        MOV #1,-(SP)
        EMT 42
        MOV @#BUFFER+2,-(SP)
        MOV #BUFF1,-(SP)
        MOV #1,-(SP)
        EMT 42
        MOV @#BUFFER+4,-(SP)
        MOV #BUFF2,-(SP)
        MOV #1,-(SP)
        EMT 42
```

```
        MOV @#BUFFER+6,-(SP)            ;CONVERT UIC TO ASCII
        MOV #BUFF3,-(SP)
        MOV #5,-(SP)
        EMT 42
        MOV @#BUFFER+10,-(SP)           ;AND PROTECT CODE
        MOV #BUFF4,-(SP)
        MOV #5,-(SP)
        EMT 42
        MOV #BUFOUT,-(SP)               ;WRITE OUT LINE OF LABEL
        MOV #LINK,-(SP)
        EMT 2
        MOV #LINK,-(SP)
        EMT 1
        BR START1
ERROR:  EMT 60
        .WORD 0
LINK:   .WORD 0             ;LINK BLOCK1
        .RAD50 /DEV/
        .WORD 0,0
        .WORD 0,2
FILE:   .WORD 0,0,0,0,0
BUFFER: .WORD 0,0,0,0
        .WORD 0,0,0,0
BUFOUT: .WORD 36
        .WORD 0
        .WORD 34
BUFF:   .BYTE 0,0,0
BUFF1:  .BYTE 0,0,0,56
BUFF2:  .BYTE 0,0,0,40,40
BUFF3:  .WORD 0,0,0
        .BYTE 40,40
BUFF4:  .WORD 0,0,0
        .BYTE 15,12,0,0
        .END START
```

```
        .TITLE MTFMAT
;
;       FORMAT MAG TAPE BY WRITING 2 TAPE
;       MARKS AT BEGINNING
START:  MOV #2,@#177540
        CLR @#177550
        TSTB @#177540
        BPL .-4
        MOV #5,@#177540
        TSTB @#177540
        BPL .-4
        MOV #401,@#177550
        MOV #5,@#177540
        EMT 60
        .END START
```

NAME: SO:

DESCRIPTION - SO: is a device driver capable of sending information to a storage tube display unit hooked to a DR11-A interface unit. The information may be either text or graphics. For text output data lines are drawn until either 38 lines are output or a form feed is encountered. To continue the user must push the 'PAGE' button on the display. The screen is erased and the next line of characters started. In text mode tabs are converted to spaces and illegal characters are ignored. For graphics output the user must set a special function code to 1. This causes SO: to ignore any normally illegal characters.

USES - SO: replaces the teletype in most look and see applications, i.e. when a hard copy is not required. The effective character speed, being about 50 K.Baud, allows more effective use of the DOS system for development purposes. In particular, the Editor and PIP programs almost require SO: as an extension of the keyboard service.

PROGRAMMING - SO: is just another device in the system library and can be employed in the same manner as KB:, PP:, etc. Its one limitation is that only one dataset can be opened at a time. For graphics SO: must be called from a user program. This is because the setting of a function code is not possible from the system programs. For further information on DOS I/O programming and special function codes, the reader should consult Chapter 2 of the DOS Users Guide. However, the function code of SO: is initialized to Ø for text mode and the user normally would not change it.

ERRORS - no errors are generated by SO:

- A listing of SO: follows: -

```
R0=%0
R1=%1
R5=%5
SP=%6
        .GLOBL SO
        .TITLE SO
;
;       ASCII OUTPUT DISPLAY HANDLER
;
;       ALLOWS 32 LINES THEN WAITS FOR ACTION ON CONSOLE
;
SO:     .WORD 0             ;START OF DRIVER TABLE
        .WORD 672           ;TERMINAL,ASCII,BIN,OUTPUT,OPEN,SPECIAl
        .BYTE 4             ;128 BYTE BUFFER
        .BYTE SO.INT-SO
        .BYTE 200           ;PRI =4
        .BYTE SO.OPN-SO
        .BYTE SO.TRN-SO
        .BYTE 0
        .BYTE SO.SPC-SO ;SPECIAL FUNCTIONS
        .BYTE 0
        .RAD50 /SO/                 ;NAME OF DEVICE
;
;  DATA AREA
;
BYTCNT: .WORD 0
NBYTE:  .WORD 0
LENGTH: .BYTE 0
LINES:  .BYTE 0
STORE2: .WORD 0
SWITCH: .BYTE 0
        .EVEN
;
;       SPECIAL FUNCTION  0 CHECK FOR SPACES AND FORM FEEDS
;                         1 NO CHECKS
;
SO.SPC: MOV SO,R0                   ;GET DDB ADDRESS
        MOV @2(R0),CODE             ;GET SPECIAL CODE
        TST     (SP)+
        JMP     @14(R0)
CODE:   .WORD 0
;
;       OPEN ROUTINE RESET AND HOME ERASE
;
SO.OPN: MOV #14,@#177532
        CLR @#177534                ;CLEAR INPUT BUFFER
        MOVB #332,LINES             ;SET LINE COUNTER NOW
SOT10:  TSTB @#177530
        BPL     SOT10
        MOV SO,R0           ;GET ADDRESS OF DDB
        TST (SP)+           ;GET RID OF CALL IN STACK
        JMP     @14(R0) ;COMPLETED RETURN CALL
;
;       TRANSFER ROUTINE  SET UP EVERYTHING
;               LINE COUNT
;               WORD COUNT AND BYTE POINTERS
;       AND SET DISPLAY IN MOTION
```

```
;
SO.TRN: MOV SO,RØ                ;GET DDB POINTER
        MOV 6(RØ),NBYTE          ;GET POINTER TO DATA
        MOV 1Ø(RØ),R1            ;AND WORD COUNT
        ASL R1                   ;MULTIPLY BY TWO
        MOV R1,BYTCNT    ;KEEP IT
;
;       ALTER STACK SETUP TO FAKE AN INTERRUPT
;       THIS PERMITS A RETURN "RTI" LATER ON
;
        MOV (SP),-(SP)           ;MOVE UP 1 WORD
        MOV @#177776,2(SP)       ;SET "INTERRUPTED" PS
;
;       INTERRUPT ROUTINE
;
SO.INT: BIC #1ØØ,@#177534        ;DISABLE INTERRUPT
        TST BYTCNT               ;TEST COUNT FOR END
        BPL SO.DUN               ;FINISHED
        TST CODE
        BNE     SOT1     ;NO TAB OR FF CHECK
        TSTB SWITCH              ;TAB IN PROGRESS ?
        BNE SOT2Ø                ;YES BRANCH
        CMPB #177,@NBYTE                 ;IGNORE RUBOUT
        BEQ SOT2
        BITB #14Ø,@NBYTE         ;CONTROL ???
        BNE SOT1                 ;NO WRITE
        CMPB #13,@NBYTE          ;V TAB ?
        BEQ SOT3                 ; YES !
        CMPB #15,@NBYTE          ;CARRIAGE RETURN ?
        BEQ SOT1
        CMPB #14,@NBYTE          ;FORM FEED?
        BEQ SOT3                 ;YES WAIT
        CMPB #12,@NBYTE          ;LINE FEED
        BEQ     SOT4             ;YES
        CMPB #11,@NBYTE          ;NEXT CHAR A TB?
        BNE     SOT2             ;NO TAB
        INCB SWITCH              ;SET TAB SWITCH
SOT2Ø:  MOVB #4Ø,@#177536
        INCB LENGTH              ;END
        BITB #7,LENGTH           ;TEST TO SEE IF FIRST
        BEQ SOT21                ;3 BITS ARE ZERO
        BIS #1ØØ,@#177534        ;NOT ZERO
        RTI
SOT21:  CLRB SWITCH              ;FINISHED TAB RESET SWITCH
        BR SOT5Ø
SOT1:   MOVB @NBYTE,@#177536     ;SEND CHAR
        INCB LENGTH              ;ONE LESS TAB SPACE
SOT5Ø:  BIS #1ØØ,@#177534        ;ENABLE INTERRUPT
SOT5:   INC NBYTE
        INC BYTCNT
        RTI
;
;
SO.DUN: MOV @#44,-(SP)           ;R.RSAV ADDRESS POINTER
        JSR R5,@(SP)+            ;SAVE REGISTERS FOR RETURN
        MOV SO,RØ                ;GET DDB POINTER INTO RØ
        JMP     @14(RØ)          ;RETURN TASK DONE
```

```
;
;       FORM FEED INSERT LINE FEED AND BOMB LINE COUNTER
;       THEN DROP TO LINE FEED ROUTINE
;
SOT3:   MOVB #12,@NBYTE
        CLRB LINES
;
;       LINE FEED FOUND CHECK FOR END OF PAGE
;       AND RESET VARIOUS PLACES
;
SOT4:   MOVB #377,LENGTH                ;RESET LENGTH
        INCB LINES
        BMI SOT1                ;NOT FULL PAGE YET
;
;       END OF PAGE  WAIT FOR "PAGE" FROM CONSOLE
;
        MOVB #332,LINES
        MOV #140,@#177776       ;LOWER PRIORITY TO ALLOW TTY IN
SOT12:  TST @#177530            ;ANY ACTION
        BPL     SOT12           ;NO
        CMPB #14,@#177534
        BNE SOT12               ;NOT CORRECT CHAR    WAIT
        CLRB @#177534
        MOV #200,@#177776       ;RESET PRIORITY
        MOV #14,@#177532        ;ERASE SCREEN
SOT13:  TSTB @#177530
        BPL SOT13
        CLRB LENGTH
SOT2:   INC NBYTE
        INC BYTCNT
        JMP SO.INT              ;CONTINUE ON
        .END
```

TR: Pseudo-Device Driver for the shared memory between 2 PDP-11 Processors.

DESCRIPTION - TR: passes information between 2 DOS systems using a shared memory block. 256 words are transferred between the CPU's at memory speeds. One extra memory word at the start of the buffer controls the transfers. This word is initially zero but is set when data is available. Reading of the data will then occur with this word being cleared. An end of file bit controls the final transfer.

USES - Immediate file transfers between 2 CPU'S.

PROGRAMMING - A common core box is required. TR: is programmed as any other non-file DOS device. 257 words are needed from the common core.

ERRORS - no errors are generated from TR:

- A listing of TR follows -

```
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
TRCSR=140000
TRBUF=140002
;
;       TRANSFER ROUTINE FOR COMMON CORED CPU'S
;
        .TITLE TR
        .GLOBL TR
TR:     .WORD 0
        .WORD 336       ;IN OUT BIN ASC ONLY
        .WORD 4 ;NO INT ROUTINE-64 WORDS XFER
        .BYTE 0
        .BYTE TR.OPN-TR
        .BYTE TR.TRN-TR
        .BYTE TR.CLS-TR
        .WORD 0
        .RAD50 /TR /
;
;       TRAN ROUTINE NO INTS
;
TR.TRN: MOV TR,R0
        MOV 10(R0),R2
        BIT #4,12(R0)
        BNE TRIN
;
;       SET VALUES FOR OUTPUT
;
TROUT:  MOV 6(R0),R1
        MOV #TRBUF,R3
        CLR R4
        MOV #1,R5
        TST     @#TRCSR         ;WAIT FOR CLEAR CSR
        BNE     TROUT   ;BEFORE CONTINUING
        BR LOOP
;
;SET VALUES FOR INPUT
;
TRIN:   MOV 6(R0),R3
        MOV #TRBUF,R1
        CLR R5
        MOV #1,R4
;
;       LOOP HERE FOR DONE
;
LOOP:   TST @#TRCSR
        BMI TREND
        CMP R4,@#TRCSR
        BNE LOOP
;
```

```
;       TRANSFER DATA IN OR OUT
;
LOOP1:  MOV (R1)+,(R3)+
        INC R2
        BMI LOOP1
        MOV R5,@#TRCSR
LOOP3:  MOV     (SP)+,R5
        MOV     @#177776,-(SP)
        MOV     R5,-(SP)
        SUB     #14,SP
        JMP @14(R0)
;
;
TREND:  MOV 10(R0),16(R0)
        CLR     @#TRCSR         ;CLEAR FOR FURTHER OUTPUT
        BR LOOP3
;
;       CLOSE ROUTINE  IGNORE IF INPUT MODE
;
TR.CLS: BIT #4,12(R0)
        BNE LOOP3
        TST     @#TRCSR         ;WAIT FOR CLEAR CSR
        BNE     TR.CLS          ;BEFORE CLOSING
        MOV #100000,@#TRCSR
        BR LOOP3
TR.OPN: MOV     #TRCSR,R1       ;TRCSR ADDR
        BIC     #100001,(R1)    ;CLEAR UNWANTED
        ADD     #100,(R1)
        TSTB    (R1)            ;MAKE SURE BOTH HERE
        BPL     .-2
        SUB     #100,(R1)       ;CLEAR
        BR      LOOP3
        .END
```