```
{PASCAL PROGRAM TO MAKE CONCEPT LOOK LIKE A TERMINAL :
        IT READS FROM THE KEYBOARD (USER INPUT)
                AND WRITES TO THE DATACOM2 PORT
        IT READS FROM THE DATACOM2 PORT
                AND WRITES TO THE CONSOLE
}
PROGRAM TTY;


CONST

        KBRD   =    35;    {KEYBOARD UNIT }
        DCOM   =    32;    {DATACOMM UNIT }
        CONS   =    1;     {CONSOLE  CRT }
VAR
        CTRLC :CHAR;   { ETX }
        CH:PACKED ARRAY[0..1] OF CHAR;          {CHARACTER THAT IS EITHER READ OR
WRIT}
        baudrate:integer;

      BEGIN
        writeln;
        write('Enter Baud Rate Code: '); readln(baudrate);
        unitstatus(dcom,baudrate,2);

        CTRLC := CHR(3);

        REPEAT
                IF UNITBUSY(DCOM)
                        THEN BEGIN
                                UNITREAD(DCOM,CH,1);       {READ A CHAR}
                                UNITWRITE(CONS,CH,1);        {WRITE TO SCREEN}
                        END;

                IF UNITBUSY(KBRD)
                        THEN BEGIN
                                UNITREAD(KBRD,CH,1);
                                UNITWRITE(DCOM,CH,1);
                        END;
        UNTIL  CH[0] = CTRLC;
END.
```

```
  page
; EQUATES FOR ALL DATACOM DRIVER SOFTWARE
; BIT NUMBER DEFINITIONS
;
BITD0       EQU         0                       ;BIT 0
BITD1       EQU         1                       ;BIT 1
BITD2       EQU         2                       ;BIT 2
BITD3       EQU         3                       ;BIT 3
BITD4       EQU         4                       ;BIT 4
BITD5       EQU         5                       ;BIT 5
BITD6       EQU         6                       ;BIT 6
BITD7       EQU         7                       ;BIT 7
;
; Buffer control table INTERNAL Flag  bit definitions **LO BYTE***  BF_INTL
; Low order byte
;
SAVSR       EQU         BITD0                   ;SET=SAVED ENTRY SR
ENQFLG      EQU         BITD4                   ;SENT ENQ WAITING FOR ACK
;
; Internal Flag masks
;
BUFFLGM     EQU         1                       ;CLEAR ALL BUT SAVSR
;
; Buffer Control Table PROTOCOL flag bit definitions  ****lo byte**** BF_PROF
;
LINE        EQU         BITD0                   ;LINE TYPE HANDSHAKE
XONXOFF     EQU         BITD1                   ;XON/XOFF HANDSHAKE
ENQACK      EQU         BITD2                   ;ENQ/ACK HANDSHAKE
CTSLIN      EQU         BITD3                   ;LINE IS CTS
DSRLIN      EQU         BITD4                   ;LINE IS DSR
DCDLIN      EQU         BITD5                   ;LINE IS DCD
INVBUSY     EQU         BITD6                   ;1=LINE IS INVERTED(O) WHEN BUS
Y
DATACOM     EQU         BITD7                   ;DATACOM 1 WHEN SET ELSE DCO
;
; BUFFER CONTROL TABLE PROTOCOL FLAG BIT DEFINITIONS ***HI BYTE***BF_PROF
;
PROT_P2     EQU         BITD0                  ;IF SET THEN SOME TYPE OF PROTOC
OL EXISTS
                                               ;ELSE NO PROTOCOLS --BUFFERS OVE
RFLOW ETC
MODM_P2     EQU         BITD1                  ;IF SET THEN A MODEM PROTOCOL EX
ISTS
NMOD_P2     EQU         BITD2                  ;IF SET THEN NULL MODEM PROTOCOL
(PROBABLY OF LITTLE USE)
FULL_P2     EQU         BITD3                  ;IF SET THEN FULL DUPLEX (DERFAL
LT)
;                                              ;OTHERWISE HALF DUPLEX
;
;
;       WRITE BUFFER flag word bit definitions    FLAG 1 ->>lo byte
;
ATHI_W1     EQU         BITD0                  ;AT HI WATER MARK
ATLO_W1     EQU         BITD1                  ;AT LOW WATER MARK
BGHI_W1     EQU         BITD2                  ;ABOVE HI WATER MARK
LTLO_W1     EQU         BITD3                  ;BELOW LOW WATER MARK
OUTE_W1     EQU         BITD4                  ;IF 0 DATA FROM BUFFER TO PORT I
NABLED
INPE_W1     EQU         BITD5                  ;IF 0 DATA FROM USER TO BUFFER I
NABLED
```

```
                                                    ;OTHERWISE CONTROLLED INTERNALLY
INPC_W1     EQU         BITD7                       ;IF SET(1) THEN USER IS CONTROLL
ING INPE
;
;      WRITE BUFFER flag word bit definitions       FLAG 2 ->>lo byte
;
FULL_W2     EQU         BITD0                       ;IF SET (1) THEN BUFFER IS FULL
EMPT_W2     EQU         BITD1                       ;IF SET (1) THEN BUFFER IS EMPTY
LOST_W2     EQU         BITD2                       ;DATA LOST ON INPUT (USER OVERRU
NS BUFFER)
AULF_W2     EQU         BITD3                       ; IF SET THEN AUTOMATIC ADD IF L
F AFTER CR
CRTF_W2     EQU         BITD4                       ;IF SET THEN PREVIOUS CHARACTER
WAS CR
;
;      READ BUFFER flag word bit definitions        FLAG 1 ->>LO BYTE
;
ATHI_R1     EQU         BITD0                       ;AT HI WATER MARK
ATLO_R1     EQU         BITD1                       ;AT LOW WATER MARK
BGHI_R1     EQU         BITD2                       ;ABOVE HI WATER MARK
LTLO_R1     EQU         BITD3                       ;BELOW LOW WATER MARK
OUTE_R1     EQU         BITD4                       ;IF 0 DATA FROM PORT TO BUFFER E
NABLED
INPE_R1     EQU         BITD5                       ;IF 0 DATA FROM BUFFER TO USER E
NABLED
OUTC_R1     EQU         BITD6                       ;IF SET(1) THEN USER IS CONTROLL
ING OUTE
                                                    ;OTHERWISE CONTROLLED INTERNALLY
INPC_R1     EQU         BITD7                       ;IF SET(1) THEN USER IS CONTROLL
ING INPE
;
;      READ BUFFER flag word bit definitions
;
FULL_R2     EQU         BITD0                       ;IF SET (1) THEN BUFFER IS FULL
EMPT_R2     EQU         BITD1                       ;IF SET (1) THEN BUFFER IS EMPTY
LOST_R2     EQU         BITD2                       ;DATA LOST ON INPUT (PORT OVERRU
NS BUFFER)
;
; Table flag's masks
;
DCMFLGM     EQU         $80                          ;LEAVE DATACOM UNTOUCHED
;
; 68000 Interrupt Auto Vector Addresses
;
VEC1        EQU         $64                          ;AUTO VECTOR #1-DATA COM CONTRO
L
                                                     ;This is the VIA used in line
                                                     ;protocols
VEC2        EQU         $68                          ;AUTO VECTOR #2-DC 1
VEC4        EQU         $70                          ;AUTO VECTOR #4-DC 0
;
;
;*************************************************************************
;
; Unit I/O Command codes  --found IN D4.W
;
INSTCMD     EQU         0                            ; Install the unit
READCMD     EQU         1                            ; read command
WRCMD       EQU         2                            ; write command
CLRCMD      EQU         3                            ; CLEAR THE UNIT
BUSYCMD     EQU         4                            ; busy command
STSCMD      EQU         5                            ; STATUS COMMAND -ACTUAL COMMA
DS IN D2.W
UNMCMD      EQU         6                            ; unmount command
```

```
;
D2_FREEW    EQU         1                   ;RETURN WRITE BUFFER FREE SPACE
D2_FREER    EQU         2                   ;RETURN READ BUFFER FREE SPACE
D2_BAUDS    EQU         3                   ;SET READ/WRITE BAUD RATE
D2_PARTY    EQU         4                   ;SET PARITY
D2_PORT     EQU         5                   ;SET DATA COM PORT (0 OR 1)
D2_CHARS    EQU         6                   ;SET CHARACXTER SIZE
D2_HANDS    EQU         7                   ;SET HANDSHAKE METHOD
D2_WRTHI    EQU         8                   ;SET WRITE HI WATER MARK(NUMBER
OF CHARACTERS LEFT FREE)
D2_WRTLO    EQU         9                   ;SET WRITE LO WATER MARK(NMBR CH
ARS LFT IN BUFFER)
D2_REAHI    EQU         10                  ;SET READ HI WATER MARK(NMBR OF
CHARS LEFT FREE)
D2_REALO    EQU         11                  ;SET READ LO WATER MARK(NMBR OF
CHARACTERS LFT IN BFR)
D2_RESTS    EQU         12                  ;RETURN READ STATUS
D2_WRSTS    EQU         13                  ;RETURN WRITE STATUS
D2_TBLALL   EQU         14                  ;RETURN TABLE STATE OF BUFR CTL,
 READ, AND WRITE
D2_BFCTRL   EQU         15                  ;RETURN STATE OF BUFFER CONTROL
TABLE
D2_BFWRT    EQU         16                  ;RETURN STATE OF WRITE CONTROL T
ABLE
D2_BFRED    EQU         17                  ;RETURN STATE OF READ CONTROL TA
BLE
D2_OUTRD    EQU         18                  ;USER DISABLE OF OUTBOARD READ(D
EVICE DISABLE)
D2_INBRD    EQU         19                  ;USER DISABLE OF INBOARD READ(BU
FFER DISABLE)
D2_OUTWT    EQU         20                  ;USER DISABLE OF OUTBOARD WRITE
(DEVICE DISABLE)
D2_INBWT    EQU         21                  ;USER DISABLE OF INBOARD WRITE (
BUFFER DISABLE)
D2_WBCHR    EQU         22                  ;RETURN THE NUMBER OF CHARACTERS
 IN WRITE BUFFER
D2_RBCHR    EQU         23                  ;RETURN THE NUMBER OF CHARACTERS
 IN READ BUFFER
;
;
;           THE ABOVE IS WILD AND WOOLY AND MAY BE OF LITTLE US TO A SIMPLE
;           HIGHER LEVEL PROTOCOL-- HOWEVER THE HIGHER YOU GET THE MORE USE
;           SOME OF THESE REPORTING FUNCTIONS MAY BE
;
;
;
;*****************************************************************************
;
; 68000 status Register values
;
;
UPRMSK      EQU         $A000                ; KEEPS ALL STATES AND TRACE B
TS AS IS
STATEMSK    EQU         $2000                ; ANDS OFF STATE BIT
TRACEMSK    EQU         $8000                ; ANMDS OFF TRACE BIT
INTMSK      EQU         $0700                ; ANDS OFF ALL INT LEVELS
INT4        EQU         $400                 ; INTERRUPT LEVEL 4 AND LOWER
INT2        EQU         $200                 ; ETC LEVEL2
INT1        EQU         $100                 ;ETC LEVEL 1
DISINT4     EQU         $2400                ; Disable all DataCom 0 (prior
ty 4) and below device ints
DISINT2     EQU         $2200                ; DISABLE DATACOM 1 (priority
 )and below ints
DISINT1     EQU         $2100                ; Disable DataCom-Control int
```

```
; VIA Addresses
;
ORA         EQU         $30F63                  ; PORT A
DDRA        EQU         $30F67                  ; PORT A DATA DIRECTION REG.
NHIRA       EQU         $30F7F                  ; PORT A W/O HANDSHAKE(IGNORE D
DRA)
;
  VIA register values
;
IODDRA      EQU         $80                     ; PORT A BIT CONFIGURATION
;
;*************************************************************************
;
; UART register definitions
;
UARTDC0     EQU         $30F20                  ;BASE ADDRESS OF DATACOM 0 UART
DC1OFF      EQU         $20                     ;OFFSET FROM DC0 BASE TO DC1 BA
SE
DATAREG     EQU           1                     ;DATA PORT REGISTER INDEX
STATRI      EQU           3                     ;STATUS REGISTER INDEX
CMDREGI     EQU           5                       ;COMMAND REGISTER INDEX
CTLREGI     EQU           7                       ;CONTROL REGISTER INDEX
;
;    UART STATUS REGISTER EQUATES
;
S_PARI      EQU         BITD0                   ;PARITY ERROR IF SET--SELF CLEAR
ING
S_FRAME     EQU         BITD1                   ;FRAMING ERROR IF SET --SELF CLE
ARING
S_OVRN      EQU         BITD2                   ;DATA OVERRUN IF SET
S_RCVF      EQU         BITD3                   ;RECEIVE REGISTER FULL IF SET -C
LEARED BY READ DATA
S_WRTE      EQU         BITD4                   ;WRITE REGISTER EMPTY IF SET
S_DCD       EQU         BITD5                   ;DATA CARRY DETECT IF LO ---WIRE
D LOW
S_DSR       EQU         BITD6                   ;DATA SET READY IF LOW --- WIRED
 LOW
S_IRQ       EQU         BITD7                   ;INTERRUPT REQUEST IF SET
;
;                                   S_RCVF EQUIVALENT TO RCVBF
;                                   S_WRTE EQUIVALENT TO XMITBE
;    UART COMMAND REGISTER
;                       NOTE:cannot or members of same section together
;
CM_DISP     EQU         0                       ;DISABLE PARITY
CM_OPBT     EQU         $20                     ;ODD PARITY BOTH XMIT AND RCV
CM_EPBT     EQU         $60                     ;EVEN PARITY BOTH XMIT AND RECE
VE
CM_MPBD     EQU         $A0                     ;MARK PARITY BIT UPON XMIT -PAR
TY CK DISABLED
CM_SPBD     EQU         $E0                     ;SPACE PARITY BIT ON XMIT - PAR
TY CK DISABLED
;-------------------------------------
CM_ECHO     EQU         $10                     ;IF SET-ECHO MODE FOR RECEIVER
;-------------------------------------
CM_DTRL     EQU         $1                      ;ENABLE RCVR/XMITRR IF SET DTR
AR=LOW
;-------------------------------------
CM_IRQD     EQU         $2                      ;DISABLE INTERRUPTS IF SET ---
OTE CORVUS CUTEY
                                                ;  THIS IS ENABLED FROM STATUS
IT 3, NOT BIT 0
                                                ;  AS IS INDICATED IN SYNERTEK
ITERATURE
```

```
CM_TELO     EQU         $4                          ;XMIT ENABLED RTS BAR LO
CM_TDLO     EQU         $8                          ;XMIT DISABLED RTS BAR LO
CM_TDBRK    EQU         $C                          ;XMIT DISABLED --XMIT BREAK
;
;                       SOME USEFUL MACRO COMMANDS
;                       FOR THE COMMAND REGISTER
TURNOFF     EQU         CM_IRQD
XMITENB     EQU         CM_TELO
XMITDIS     EQU         CM_TDLO
CMDRC       EQU         CM_DTRL+CM_TDLO             ;NO XMIT INT, RCV INT,ENAB DTR,
NO PARITY
CMDRWC      EQU         CM_DTRL+CM_TELO             ;SAME AS CMDRC XCEPT XMIT INTERR
UPTS ENABLED ALSO
;
;---------------------------------------
;
CLRD3D2     EQU         $F3                          ;CLEAR BITS D3 & D2 A MASK
;
;=================================================================
;
;    UART CONTROL REGISTER EQUATES
;
;           NOTE: Baud is lower 4 bits of control word--see BAUDCNV table below
;
CR_STPB     EQU         $80                         ;IF 0 THEN = 1 STOP BIT
                                                    ;  IF SET AS INDICATED = 2 STOP
BITS IF NO PARITY
                                                    ;                       =1 STOP B
IT IF 8 BIT CHAR + PARITY
                                                    ;                       =1.5 STOP
 BITS IF 5BIT WORD NO PARITY
;---------------------------------------
CR_WRDL8    EQU         0                           ;8 BITS WORD LENGTH
CR_WRDL7    EQU         $20                         ;7 BIT WORD LENGTH
CR_WRDL6    EQU         $40                         ; 6 ETC
CR_WRDL5    EQU         $60                         ; 5 ETC.
;---------------------------------------
CR_EXTCLK   EQU         0                           ;EXTERNAL RECEIVE CLOCK
CR_BDCLK    EQU         $10                         ;BAUD RATE GEN FOR CLOCK
;---------------------------------------
;
; UART CONTROL REGISTER CONSTANTS FOR UART SETUP
;
CTLRC       EQU         CR_BDCLK+CR_WRDL8            ;1 STOP BIT,8BIT WORD LENGTH,BA
UD RATE GENERATOR

; ASCII Control characters for printer control
;
XON         EQU         $11                          ;CAN XMIT (CTL-Q)
XOFF        EQU         $13                          ;STOP XMIT (CTL-S)
ENQ         EQU         $05                          ;READY FOR MORE? (CTL-E)
ACK         EQU         $06                          ;YES, I'M READY (CTL-F)
NULL        EQU         $00                          ;NULL CHARACTER-DO NOTHING
CR          EQU         $0D                          ;CARRIAGE RETURN
LF          EQU         $0A                          ;LINE FEED
;
; Maximum Parameter values for Unitstatus Set table entry functions
;
MAXBAUD     EQU         6                            ;FOR SET BAUD RATE
MAXPRTY     EQU         4                            ;FOR SET PARITY
MAXWRDS     EQU         1                            ;FOR SET WORD SIZE
MAXDTCM     EQU         1                            ;FOR SET DATACOM
MAXHNDS     EQU         7                            ;FOR SET HANDSHAKE TYPE
MAXWHI      EQU         133                         ;HI WATER WRITE MAX # CHARS
```

```
MAXRLO      EQU         80                  ;LO WATER READ #CHARS MAX
;
; error codes (IORESULT)
;
INVCMD      EQU         IOEioreq            ;invalid cmd-(invalid I/O reque
st)
INVTBLID    EQU         IOEtblid            ;invalid table id
INVPRM      EQU         IOEuiopm            ;invalid parameter
INVFNC      EQU         IOEfnccd            ;invalid function code
;
; Miscellaneous definitions
;
TRUE        EQU         1                   ; Pascal true boolean value
ON          EQU         1                   ;LISTING CONTROL - START LISTIN
G
OFF         EQU         0                   ;LISTING CONTROL - STOP LISTING
TBLSTATE    EQU         22                  ;21 POSSIBLE STATUS INQUIRIES
HILOMSK     EQU         $F0                 ;MASK OFF WATER MARKS -THRO THE
M AWAY
```

```
        page
 ;
 ; File: os.gbl.asm.text
; Date: 20-Aug-82
 ;
;
; Corvus CONCEPT operating system data structure equates
;

;
; Additional Corvus CONCEPT I/O result codes
;
IOEioreq equ    3       ; Invalid I/O request
                        ;
IOEnotrn equ    21      ; Transporter not ready
IOEtimot equ    22      ; Timed out waiting for Omninet event
IOEnobuf equ    23      ; Read without a valid write buffer
                        ;
IOEwndfn equ    32      ; Invalid window function
IOEwndbe equ    33      ; Window create boundary
IOEwndcs equ    34      ; Invalid character set
IOEwnddc equ    35      ; Delete current window
IOEwndds equ    36      ; Delete system window
IOEwndiw equ    37      ; Inactive window
IOEwndwr equ    38      ; Invalid window record
IOEwndwn equ    39      ; Invalid system window number
                        ;
IOEnodsp equ    40      ; Display driver not available
IOEnokyb equ    41      ; Keyboard driver not available
IOEnotim equ    42      ; Timer driver not available
IOEnoomn equ    43      ; OMNINET driver not available
IOEnoprt equ    44      ; Printer driver not available
IOEnfdrv equ    45      ; No floppy drive at slot
                        ;
IOEtblid equ    50      ; Invalid table entry ID
IOEtblfl equ    51      ; Table full
IOEtbliu equ    52      ; Table entry in use
IOEkybte equ    53      ; Keyboard transmission error
IOEuiopm equ    54      ; Invalid unit I/O parameter
IOEprmln equ    55      ; Invalid parameter block length
IOEfnccd equ    56      ; Invalid function code
IOEclkmf equ    57      ; Clock (hardware) malfunction

;
 ; System Common Pointer
;
pSysCom    equ  $0180   ;pointer to address of SYSCOM
SysKybdFlg equ  $0184   ;keyboard control flags
SysByteScn equ  $0186   ;display driver - bytes per scan line

;
 ; System Common Equates
;
SCiorslt equ    0       ;word - I/O result
SCprocno equ    2       ;word - current process number
SCfreehp equ    4       ;lint - free heap pointer
SCjtable equ    8       ;lint - jump table pointer
SCsysout equ    12      ;lint - default output file pointer
SCsysin  equ    16      ;lint - default input file pointer
```

```
SCutable equ     28       ;lint - user table pointer
SCtoday  equ     32       ;word - system date
SCcodejt equ     34       ;lint - code jump table pointer
SCnxtpro equ     38       ;word - next process number
SCnumpro equ     40       ;word - number of processes
SCprotbl equ     42       ;lint - process table pointer
SCbootnm equ     46       ;lint - boot device name pointer
SCmemmap equ     50       ;lint - memory map pointer
SCbootdv equ     54       ;word - boot device number
                          ;
8; CONCEPT additions
;        equ     56       ;word - unused
;        equ     58       ;word - unused
SCslttbl equ     60       ;lint - slot table pointer
SCrootw  equ     64       ;lint - root window record pointer
SCcurrw  equ     68       ;lint - current window record pointer
SCcurrk  equ     72       ;lint - current keyboard record pointer
SCuserid equ     76       ;word - Constellation user ID
SCvrsnbr equ     78       ;lint - current version number string pointer
SCvrsdat equ     82       ;lint - current version date string pointer
SCwndtbl equ     86       ;lint - window table pointer
SCsusinh equ     90       ;word - suspend inhibit count
SCsusreq equ     92       ;word - suspend request if non-zero

        page
;
 ; System Vector Equates
;
SVuwrite equ      0*4     ;unit write
SVuread  equ      1*4     ;unit read
SVput    equ      4*4     ;put
SVget    equ      5*4     ;get
SVinit   equ      6*4     ;init
SVopen   equ      7*4     ;open
SVclose  equ      8*4     ;close
SVwrchar equ      9*4     ;writechar
SVrdchar equ     10*4     ;readchar
SVblkio  equ     11*4     ;blockio
SVnew    equ     13*4     ;new
SVmark   equ     15*4     ;mark
SVrlease equ     16*4     ;release
SVmavail equ     17*4     ;memory available
SVgetdir equ     18*4     ;get directory
SVcrkpth equ     24*4     ;crack path name
SVcli    equ     31*4     ;command line interpreter
SVgetvnm equ     32*4     ;get volume names
SVvaldir equ     33*4     ;check valid directory
SVflpdir equ     34*4     ;flip directory
SVschdir equ     35*4     ;search directory
SVdelent equ     36*4     ;delete directory entry
SVputdir equ     37*4     ;write directory
SVuinstl equ     38*4     ;unit install

;
; Memory Map Equates
;
MMlodta  equ     0        ;lint - low data pointer
MMhidta  equ     4        ;lint - high data pointer
MMlocod  equ     8        ;lint - low code pointer
MMhicod  equ     12       ;lint - high code pointer
MMbtsw   equ     16       ;word - boot switches
MMbtdev  equ     18       ;word - boot device number
MMbtslt  equ     20       ;word - boot slot number
MMbtsrv  equ     22       ;word - boot server number
```

```
        page
;
 ; Unit Table Equates
;
UTiodrv equ     2       ;lint - I/O driver pointer
UTblf   equ     6       ;bool - blocked device flag
UTmtd   equ     7       ;bool - mounted device flag
UTdid   equ     8       ;str7 - device ID
UTsiz   equ     16      ;lint - device size
UTslt   equ     20      ;byte - device slot
UTsrv   equ     21      ;byte - device server
UTdrv   equ     22      ;byte - disk drive nmbr
UTtyp   equ     23      ;byte - disk drive type
UTspt   equ     24      ;byte - sectors per track
UTtps   equ     25      ;byte - tracks per side
UTro    equ     26      ;bool - device read only
;       equ     27      ;byte - ... unused
UTblk   equ     28      ;lint - disk base block
UTlen   equ     32      ;       entry length

;
 ; Slot Table Equates
;
STbtslt  equ    0       ;boot slot number
STbtsrv  equ    2       ;boot server number
 STacslt  equ    4       ;active slot number
 STacsrv  equ    6       ;active server number
STalslt  equ    8       ;alternate slot number
STalsrv  equ    10      ;alternate server number
 STinfo   equ    12      ;array [1..5] of ....
                         ;
 STnmbr   equ    0       ;   slot number (1-5)
 STtype   equ    1       ;   device type (slottypes)
 STndrv   equ    2       ;   number of drives
 STinfoL  equ    4       ;   device info length

        page
;
 ; Character Set Record Equates
;
 CStblloc equ    0       ;character set data pointer
CSlpch   equ    4       ;scanlines per character (assume wide)
CSbpch   equ    6       ;bits per character (vertical height)
CSfrstch equ    8       ;first character code - ascii
CSlastch equ    10      ;last character code - ascii
CSmask   equ    12      ;mask used in positioning cells
CSattr1  equ    16      ;attributes
8;  bit 0 = 1 - vertical orientation
CSattr2  equ    17      ;currently unused

;
 ; Window Record Equates
;
WRcharpt equ    0       ;character set pointer
WRhomept equ    4       ;home (upper left) pointer
WRcuradr equ    8       ;current location pointer
WRhomeof equ    12      ;bit offset of home location
WRbasex  equ    14      ;home x value, relative to root window
WRbasey  equ    16      ;home y value, relative to root window
WRlngthx equ    18      ;maximum x value, relative to window (bits)
WRlngthy equ    20      ;maximum y value, relative to window (bits)
WRcursx  equ    22      ;current x value (bits)
WRcursy  equ    24      ;current y value (bits)
```

```
WRgrorgy equ     30      ;graphics - origin y (bits relative to home loc)
WRattr1  equ     32      ;attributes
                          ;
 invrse   equ     0       ;   inverse video mode
 undscr   equ     1       ;   underscore mode
 insmod   equ     2       ;   insert mode
viddeflt equ     3       ;   0 = W on B,        1 = B on W
noautolf equ     4       ;   0 = auto LF w/CR, 1 = no auto LF
syswin   equ     5       ;   system defined window
active   equ     6       ;   active window
 suspend  equ     7       ;   suspended window
                          ;
WRattr2  equ     33      ;attributes
                          ;
 vert     equ     0       ;   1 = vertical,    0 = horizontal screen
graphic  equ     1       ;   1 = graphics,    0 = character mode
curson   equ     2       ;   1 = cursor on,   0 = cursor off
 invcurs  equ     3       ;   1 = inverse,     0 = underline cursor
 wrapon   equ     4       ;   1 = wrap,        0 = clip at eoln
noscroll equ     5       ;   1 = no scroll,   0 = scroll
clrsc    equ     6       ;   1 = paging mode
vidset   equ     7       ;   1 = inverse      0 = normal
                          ;
WRstate  equ     34      ;used for decoding escape sequences
WRrcdlen equ     35      ;window description record length
                          ;
WRlength equ     36      ;actual window record length

         page
```

```
; DATACOM DRIVER
;
COMDRV
            BRA.S       COM001                      ;*070782* JUMP AROUND HEADER
            DATA.B      0                           ;DEVICE NOT BLOCKED
            DATA.B      31                          ;VALID CMDS - ALL VALID
            DATA.B      82,10,14,00                 ;DATE JULY 7 1982
            DATA.B      hmlen                       ;HEADER MSG LENGTH
xxx010      DATA.B      'DATACOM driver 3A DEBUG'           ;HEADER MSG
hmlen       EQU         %-xxx010
;
COM001
;           TRAP         #15
;           DATA.W       0
;           NOP
;           NOP                                     ;SETS UP DEBUGGER
            CMPI.W      #UNMCMD,D4                   ;VALID COMMAND
,BHI.S      PRNDERR                     ;NO
,MOVEM.L    D1-D6/A0-A6,-(SP)           ;SAVE REGISTERS
,CLR.L      D7                          ;CLEAR IORESULT
,MOVEA.L    D1,A3                       ;ADDRESS OF USERS BUFFER
            LEA         COMTBL,A1                   ;TURN THE COMMAND INTO A
,LSL.W      #1,D4                       ;INDEX TO THE FUNCTION
,MOVE.W     0(A1,D4.W),D4
,JSR        0(A1,D4.W)                  ;DO FUNCTION
,MOVEM.L    (SP)+,D1-D6/A0-A6           ;*** temp* for busy return in D0
,RTS
;
; Invalid Command Error
;
PRNDERR     MOVE.W      #INVCMD,D7
,RTS
;
; THE PRINTER DRIVER JUMP TABLE
;
COMTBL      DATA.W      COMINST-COMTBL              ;UNITINSTALL
            DATA.W      COMRD-COMTBL                ;UNITREAD
            DATA.W      COMWR-COMTBL                ;UNITWRITE
            DATA.W      COMCLR-COMTBL               ;UNITCLEAR
            DATA.W      COMBSY-COMTBL               ;UNITBUSY
            DATA.W      COMST-COMTBL                ;UNITSTATUS
            DATA.W      COMUNMT-COMTBL              ;UNITUNMOUNT
  ;(*$P
;*************************************************************************
**************************
;
; COMINST - UNITINSTALL ==> SETUP THE DEFAULT BUFFER CONTROL FEATURES
;   Assumes that a spurrious DataCom Control interrupt is benign and will
;   be handled by the DataCom Control interrupt service routine correctly.
;
COMINST     BSR.S       DISINTS                     ;DISABLE DATACOM INTERRUPTS
;
            LEA         BFRCTL,A0
            MOVE.B      DEFBWRT,(A0)+               ;DEFAULT WRITE BAUD RATE
            MOVE.B      DEFBRD,(A0)+                ;DEFAULT READ BAUD RATE
            MOVE.B      DEFPART,(A0)+               ;DEFAULT PARITY
            MOVE.B      DEFWRDS,(A0)+               ;DEFAULT WORD SIZE
            MOVE.W      DEFINTRN,(A0)+              ;DEFAULT INTERNAL FLAG
            MOVE.W      DEFPROT,(A0)                ;DEFAULT PROTOCOL FLAG
;
; Initialize UART from constants and Printer Control Table & Initialize VIA
;
,MOVE.B     #IODDRA,DDRA.L              ;INITIALIZE DATA DIRECTION REG FOR PORT A
,BSR.S      SETUART
```

```
,BTST       #SAVSR,(A0)                 ;ONLY RESET STATUS REG. IF
,BEQ.S      PINNOSR                     ;SAVED SR IN SAVESR1
,MOVE.W     #DISINT1,SR                 ;ALLOW ALL BUT DC CONTROL
;
; Initialize READ AND WRITE BUFFER CONTROL TABLES
;
PINNOSR     BSR         INIWRBF
            BSR         INIRDBF
;
; Setup interrupt vectors
;
            BSR         SETVECS
;
; If saved SR then restore it
;
,BSR.S      ENBINTS
,RTS
  ;(SP
;*****************************************************************************
**************************
;
; DISINTS - disable level 4 interrupts if current level is less than 4
;           If currently anything higher than lvl4 disabled dont save sr.
;           If lower than lvl14, raise to level 4 and save last value for when er
bints time.
;
DISINTS     LEA         BF_INTL+1,A0
,BCLR       #SAVSR,(A0)                 ;ASSUME NOT SAVED STATUS REG
,MOVE.W     SR,D0
            MOVE.W      D0,D5                       ; SAVE FOR MANIPS
            ANDI.W      #INTMSK,D0                  ;GET ONLY INTERRUPT LEVELS
            CMPI.W      #INT4,D0                    ;CURRENT LEVEL - LEVEL 4
;
;                                       NOTE: If 2 datcaom drivers are run simulta
neously then have to
;                                             refine this routine to distiguish le
vel based on datacom port
;
,BCC.S      DITEXIT                     ;DON'T SAVE
,BSET       #SAVSR,(A0)                 ;MARK SAVED SR
,LEA        SAVESR1,A0                  ;SAVE THE CURRENT INT STATUS
            MOVE.W      D5,(A0)
;
;                                       NOW set up disable with minimum disturban
e of upper level
;                                                 status bits --- this too wont
ork if user and
;                                                 supervisor space are both util
sed.
;
            ANDI.W      #UPRMSK,D5                  ;KEEP ONLY UPR BITS
            ORI.W       #INT4,D5                    ;LEVEL 4 DISABLE OR'D IN
            MOVE.W      D5,SR                       ;PREVENT ALL DATACOM INTERUPTS
DITEXIT     RTS
;
; ENBINTS - Restore saved SR if saved it
;
ENBINTS     LEA         BF_INTL+1,A0                ;IF SAVED SR FLAG SET
,BTST       #SAVSR,(A0)                 ;THEN RESTORE SR
,BEQ.S      EITEXIT                     ;DIDN'T SAVE SO EXIT
,MOVE.W     SAVESR1,SR
EITEXIT     RTS
;*****************************************************************************
**************************
```

```
; SETUART - Initialize UART from constants and Buffer Control Table
;
; Get UART Register Base address
;
SETUART     BSR.S       GETBASE                     ;RETURNS BASE IN A0
;
; Setup UART's Control register - index = 7  from Base
;
            MOVEQ       #CTLRC,D0                   ;1 STOP BIT,BAUD RATE GEN
            MOVE.B      BF_WRDS,D1                  ;ADD WORD SIZE-7 OR 8 BITS
            LSL.B       #5,D1                       ;MOVE INTO HI ORDER BITS
            OR.B        D1,D0                       ;00=8 BITS,01=7 BITS
            OR.B        BF_RDBD,D0                    ;ADD BAUD RATE FROM TABLE
            MOVE.B      D0,CTLREGI(A0)              ;PUT IN CONTROL REGISTER
;
; Setup UART's Command register - index = 5 from Base
;
            MOVEQ       #CMDRC,D0                   ;CMD CONSTANTS keep xmit ints
isabled
            LEA         BF_PART,A1
            MOVE.B      (A1),D1                   ;GET TABLE PARITY
            LSL.B       #5,D1                       ;PUT IN CORRECT BIT POSITION
            OR.B        D1,D0
            MOVE.B      D0,CMDREGI(A0)              ;PUT IN COMMAND REGISTER
;
; Read the Data Port and Status Register to clear all Status flags
;
            MOVE.B      DATAREG(A0),D0              ;DATA PORT AT INDEX = 1
            MOVE.B      #0,DATAREG(A0)              ;CLEAR XMIT INT
            MOVE.B      STATRI(A0),D0               ;STATUS REG AT INDEX = 3
            RTS
;-------------------------------------------------------------------
;
; GETBASE - Get address of UART's register Base address in memory
;           EXIT : (A0) = Base address
;
GETBASE     LEA         UARTDC0.L,A0                ;ASSUME USING DATACOM 0
            CLR.L       D0
            LEA         BF_PROF+1,A1                ;IF FLAG IS SET THEN MAKE
            BTST        #DATACOM,(A1)               ;D0 = THE ADDRESS OFFSET TO
            SNE         D0                          ;UART 1'S REGISTERS ELSE
            ANDI.B      #DC1OFF,D0                  ;MAKE D0 = 0
            ADDA.L      D0,A0                       ;BASE := OFFSET+UART DC0 BSE A
DR
            RTS
;*************************************************************************
***********
 ;($P
;
; SETVECS - Put interrupt routine's entry addresses into the interrupt vectors
SETVECS     LEA         DCTLINT,A0                  ;PUT DATA COM CONTROL
,MOVE.L     A0,VEC1.W                   ;INT ROUTINE IN VEC 1
,BSR.S      SETDCVEC                    ;PUT IN DATA COM XMIT/RCV RTN
,RTS
;
; SETDCVEC - Depending on which DataCom Port is being used, place the
;            address of the Xmit/Rcv DataCom int routine's entry address in
;            the interrupt vector for that DataCom.  Put a pointer to a RTE
;            instruction into the vector not being used, just in case some
;            tyrkey interrupts on that line
;
SETDCVEC    LEA         VEC4.W,A0                   ;ASSUME DC0 (VECTOR 4)
            LEA         VEC2.W,A1                   ;DC1 IS VECTOR 2
            LEA         BF_PROF+1,A2
```

```
            EXG         A0,A1                       ;NO,USE DC1 (VECTOR 2)
;
SDVUSE0     LEA         DCOMINT,A2                  ;ADDR OF XMIT/RCV INT ROUTINE
            LEA         PTRRTE,A3
            MOVE.L      A2,(A0)                     ;INTERRUPT ROUTINE
            MOVE.L      A3,(A1)                     ;PTR TO RTE
            RTS
  ;(SP
;*************************************************************************
*********************
;
; INIWRBF - Initialize Write Buffer variables to EMPTY Buffer also ENQ, BUSY ar
;           SENDLF are cleared to false.
;
INIWRBF     LEA         WRTCTL,A0                  ;WRITE BUFFER CONTROL TABLE
            LEA         WRTBUF,A1                  ;WRITE BUFFER
            MOVE.L      A1,(A0)+                    ;FILL POINTER (USED TO FILL CH
RACTERS IN)
            MOVE.L      A1,(A0)+                    ;EMPTY POINTER (USED TO EMPTY
HARACTERS OUT)
            MOVE.W      #WBFLEN,(A0)+               ;MAXIMUM SIZE OF BUFFER
            MOVE.W      #WBFLEN,(A0)+               ;NUMBER OF LOCATIONS AVAILABLE
TO FILL
            MOVE.W      #MAXWHI,(A0)+               ;NUMBER OF CHARACTERS FOR HIGH
WATER MARK
            MOVE.W      #MAXWLO,(A0)+               ;NUMBER OF CHARACTERS FOR LOW
ATER MARK
            CLR.W       (A0)                        ;RESET ALL FLAG1
            ADDQ.L      #1,A0                       ;LOW ORDER BYTE
            BSET        #LTLO_W1,(A0)                ;BELOW LOW WATER MARK
            ADDQ.L      #2,A0                      ;POINT TO FLAG2
            BSET        #EMPT_W2,(A0)                ; BUFFER IS EMPTY
            BSET        #AULF_W2,(A0)                ; AUTO LINE FEED AFTER CR IS A
SERTED
            RTS
;
;*************************************************************
; INIRDBF - Initialize READ Buffer variables to EMPTY Buffer also ENQ, BUSY and
;           SENDLF are cleared to false.
;
INIRDBF     LEA         RDCTL,A0                  ;READ BUFFER CONTROL TABLE
            LEA         RDBUF,A1                  ;READ BUFFER
            MOVE.L      A1,(A0)+                    ;FILL POINTER (USED TO FILL CH
RACTERS IN)
            MOVE.L      A1,(A0)+                    ;EMPTY POINTER (USED TO EMPTY
HARACTERS OUT)
            MOVE.W      #RBFLEN,(A0)+               ;MAXIMUM SIZE OF BUFFER
            MOVE.W      #RBFLEN,(A0)+               ;NUMBER OF LOCATIONS AVAILABLE
TO FILL
            MOVE.W      #MAXRHI,(A0)+               ;NUMBER OF CHARACTERS FOR HIGH
WATER MARK
            MOVE.W      #MAXRLO,(A0)+               ;NUMBER OF CHARACTERS FOR LOW
ATER MARK
            CLR.W        (A0)+                      ; CLEAR ENQ COUNT
            CLR.W       (A0)                        ;RESET ALL FLAG1
            ADDQ.L      #1,A0                       ;LOW ORDER BYTE
            BSET        #LTLO_R1,(A0)                ;BELOW LOW WATER MARK
            ADDQ.L      #2,A0                      ;POINT TO FLAG2
            BSET        #EMPT_R2,(A0)                ; BUFFER IS EMPTY
            RTS
;
;
;*************************************************************************
************************
```

```
; COMRD - UNITREAD
; READ FROM THE DATACOM BUFFER
;
;            INPUTS......D2 COUNT OF CHARACTERS THE USER WANTS TO READ
;                        A3 ADDRESS OF USER'S BUFFER
;
;            NOTES:    For reading, interrupts will occur when the input buffer
 full -no
;                      priming is necessary as is with writing. Also if full dupl
x activities
;                      then a read and write interrupt may be the same interrupt
have to check
;                      status flags of UART.
COMRD
;                        First see if user's count is exhausted if not-attempt a
ead
;
             TST.W       D2
             BEQ.S       COMREX                    ;COMREX GENERAL EXIT ROUTINE
;
;                        Here see if the user has disabled input in any meaningfu
 way
;
REREAD
             LEA         RB_FLG1+1,A0
             BTST        #INPE_R1,(A0)             ;IS BUFFER TO USER TRANSFER ENA
LED?
             BEQ.S       CKPORT                    ; YES IT IS -  NOW SEE IF PORT
O BUFFER XFER IS ENABLED
;
;                                    HERE SEE IF THE USER HAS DISABLED THE XFER F
OM BUFFER TO USER
             BTST        #INPC_R1,(A0)
             BEQ.S       REREAD                     ; NOPE--MACHINE DISABLED KEEP
RYING
;
;                        HERE the user has disabled his buffer input -remind him
;
             MOVE.W      #IOEirdsbl,D7
             RTS
;
;                       Here if there is any data in the buffer, give it to user.
If there is no data and
;                        the user has disabled the outboard read, remind him. How
ver if the klutz wants to
;                        read and there aint nothin there hang him in a loop wait
ng for data.
;
CKPORT
STUCKRD
             LEA         RB_FLG2+1,A0
             BTST        #EMPT_R2,(A0)             ;SEE IF BUFFER IS EMPTY
             BEQ.S       READONE                   ; NOPE GO READ A CHARACTER
             LEA         RB_FLG1+1,A0              ; BUFFER EMPTY--SEE IF INPUT IS
AT LEAST ENABLED
             BTST        #OUTE_R1,(A0)
             BEQ.S       STUCKRD                   ; YES IT IS -USER WINS A LOOP
;
;                                    Here see if the user or machine disabled the
outboard read
             BTST        #OUTC_R1,(A0)
             BEQ.S       STUCKRD                    ; MACHINE DISABLED -TRY AGAIN
;
;                        Here the user has the read port disabled ---tell him
```

```
            RTS
;
;                       Here we win the big banana -get user his characters an
anage buffer
;
READONE
            BSR.S       UGETCHR                     ;GET THE CHARACTER FOR THE US
FROM THE BUFFER
            BCS.S       RDPROB                      ;IF RDPROB THEN FOUND BY UGETC
;
;                       Here give user his character and keep track of his spac
            MOVE.B      D7,(A3)+
            SUBQ.W      #1,D2
            BRA.S       COMRD                       ;GETSMOA IF AVAILABLE
;
;                       Note that d7 is all set up to point at err if found by
etchr
;                         thus a simple return is all thats required
RDPROB
            RTS
COMREX
            MOVE.W      #IOok,D7
            RTS
;
 ;(SP
;-------------------------------------------------------------------------------
; UGETCHR  --- User level get character routine, gets the character from the r
d buffer.
;
;                       The character if gotten is returned in D7 since d0 clob
red by disints.
;                       If there are any problems, then the horr5or code is set
nto D7
;                        and the carry bit is set
;                       If all ok then carry bit is cleared
UGETCHR
            MOVEM.L     D1-D6/A0-A6,-(SP)
            BSR         DISINTS                     ;GET RID OF INTS HEREIN
            LEA         RB_EMPTY,A1                  ; A1 NOW HOLDS POINTER TO EMP
 ADDRESS
            MOVE.L      (A1),A0                     ;A0 NOW POINTS TO THE EMPTYING
OSITION OF RD BUFFER
            MOVE.B      (A0)+,D7                    ;Coming into this routine -bun
a checks made to ensure
                                                    ;   at least one character is
ailable. This then is the
                                                    ;   "get" of that character.
            MOVE.L     A0,(A1)                     ;Move the incremented pointer b
k to rb_empty
;
;           Now check for various things such as pointing beyond end of physic
 buffer
;             hi and lo water marks , and consistency of buffer sizes.
;
            LEA         RDBUF+RBFLEN,A2             ;PHYSICAL ADDRESS OF END OF BU
ER IN A2
            CMPA.L      A0,A2                       ;Note ao incremented above to
int to next logical
                                                    ; character position
                                                    ; Compare is  physical end of
ffer - next assumed
                                                    ;    character address
            BGT.S       NORWRA                      ; HERE  + OK --NO WRAP AROUND
 ADDRESS
```

```
;
            LEA         RDBUF,A0                    ;ADDRESS OF TOP OF READ BUFFER
            MOVE.L      A0,(A1)                     ; RB_EMPTY NOW RESET
;
;                       From here on out we will be screwing with things that the
 interrupt routines
;                         may well fiddle with, hence disable interrupts and hope
 for the best
;
NORWRA
            LEA         RB_FREE,A0
            ADDQ.W      #1,(A0)                     ;SINCE WE GOT CHAR, ONE MORE FRE
E SPACE
            LEA         RB_SIZE,A1
            MOVE.W       (A0),D1
            SUB.W       (A1),D1                     ; #OF FREE LOCATIONS - BUFFER SI
ZE
            BLT.S       CRWAT                       ; - OR 0 OK  HERE CHECK HI/LO WA
TER
            BGT.S       HELPRD                      ; HELPRD IS SERIOUS ERROR
;
;                       Here the number of free locations is = size of buffer, he
nce buffer is
;                        empty- reset all pointers
;
            LEA         RDBUF,A0
            LEA         RB_EMPTY,A1
            LEA         RB_FILLP,A2
            MOVE.L      A0,(A1)                      ; EMPTY ADDRESS REINITIALISED
            MOVE.L      A0,(A2)                      ; FILL POINTER REINITIALISED
            LEA         RB_FLG2+1,A0
            BSET        #EMPT_R2,(A0)
            LEA         RB_FLG1+1,A0
            BSET        #LTLO_R1,(A0)               ;SOME FLAGS RESET
            BSR       ENBINTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #0,CCR                      ;CLEAR CARRY SINCE ALL OK
            RTS
;
HELPRD                                              ; SERIOUS BUMMER BUG
            BSR       ENBINTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #IOEbszerr,D7               ;SIZING ERROR
            MOVE.W      #1,CCR                      ;SET CARRY
            RTS
;
;                       Here check the hi/lo water marks etc set flags user may s
omeday want
;
CRWAT
            BSR.S       REDWAT
            BSR       ENBINTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #0,CCR                      ; ALL OK
            RTS
;-------------------------------------------------------------------------------
----
;  redwat  -checks water marks for reading
;
;           NOTES: This routine assumes that interrupts are disabled prior
;                  to its being provoked.
REDWAT
            MOVEM.L     A0-A2/D1/D2,-(SP)
            LEA         RB_FLG1+1,A0                ;SETR UP FLAG WORD FOR MANIPS
```

```
                LEA         RB_FREE,A2
                MOVE.W      (A1),D1
                SUB.W       (A2),D1                 ; TAKE AWAY FREE SPACE, LEAVI
UMBR CHARS IN BUFFR
                LEA         RB_LOWA,A1
                CMP.W       (A1),D1                 ; HOW DOES NMBR CHARS COMPARE
 LOW WATER MARK CR-LOWA
                BEQ.S       ATLO                    ; =0 THEN AT LOW WATER
                BLT.S       BELOLO                  ; - TRHEN L;ESS THAN LOW WATE
ARK
;
;                           Here obviously + so compare with hi water marks- curre
y do not deal
;                             with the absurdity of user setting hi<lo etc.
;
                LEA         RB_SIZE,A1
                MOVE.W      (A1),D2                 ;ACTUAL SIZE OF BUFFER
                LEA         RB_HIWA,A1
                SUB.W       (A1),D2                 ;D2 NOW CONTAINS TOTAL # CHAR
O GET TO HIWAT MARK
                CMP.W       D2,D1                   ;HIWAT - ACTUAL # CHARS IN BU
                BEQ.S       ATHI                    ; AT HI WATER MARK
                BLT.S       MORHI                   ; BEYOND HI WATER MARK
                BRA         BYR
ATLO
                BSET        #ATLO_R1,(A0)
                BRA         BYR
BELOLO
                BSET        #LTLO_R1,(A0)
                BRA         BYR
ATHI
                BSET        #ATHI_R1,(A0)
                BRA         BYR
MORHI
                BSET        #BGHI_R1,(A0)
BYR
                MOVEM.L     (SP)+,A0-A2/D1/D2
                RTS
;*************************************************************************
**********************
  ;($P
; COMWR - UNITWRITE
;
;               INPUTS......D2 COUNT OF CHARACTERS THE USER WANTS TO WRITE
;                           A3 ADDRESS OF USER'S CHARACTERS
;               Setup the write buffer, and if appropriate, start filling it with
aracters.
;               NOTE:
;                     For writing, the UART has to be tricked into interrupting wh
 the xmit buffer
;                     is empty by enabling the xmit interrupt. If no xmissions the
of course its empty
;                     and it interrupts forever. Hence trickery only when sending
rst of a stream
;                     (starting interrupts) and last of a stream (stopping the lit
e dears)
;
COMWR
                TST.W       D2                      ;IS USER COUNT DONE?
                BEQ.S       COMWEX                  ;YES
;
;                           Here see if the user has disabled input in any meaning
 way
;
```

```
            BTST        #INPE_W1,(A0)             ;IS USER TO BUFFER TRANSFER ENAB
LED?(INBOARD WRITE)
            BEQ.S       CKWRTP                    ; YES IT IS -  NOW SEE IF BUFFER
 TO PORT XFER IS ENABLED
;
;                                  HERE SEE IF THE USER HAS DISABLED THE XFER FR
OM BUFFER TO USER
            BTST        #INPC_W1,(A0)
            BEQ.S       REWRITE                    ; NOPE--MACHINE DISABLED KEEP T
RYING
;
;                       HERE the user has disabled his buffer input -remind him
;
            MOVE.W      #IOEiwdsbl,D7
            RTS
;
;                      Here put the user's data into the buffer. First see if cha
r can fit in buffer, and
;                       whether or not outboard write is enabled. User may win a
wait loop given the right
;                       conditions. Also may have to prime pump -whoopeee.
;
CKWRTP
MOREWR
            LEA         WB_FLG2+1,A0
            BTST        #FULL_W2,(A0)             ;SEE IF BUFFER IS FULL
            BEQ.S       WRTONE                    ;NOPE GO WRITE A CHAR TO THE BUF
FER
            LEA         WB_FLG1+1,A0              ; BUFFER FULL - SEE IF OUTPUT IS
 AT ALL ENABLEDD
            BTST        #OUTE_W1,(A0)
            BNE.S       MCHOFF                    ; NO IT ISN'T, SEE IF MACHINE OF
 USER DIABLED
;
;                                Here see if the buffer was previously empty -if
so start xmits
;
            BSR.S       MAYWRTS                    ; HERE WE MAY START A WRITE OUT
 THE PORT
            BRA         MOREWR
;
;                                   Here see if the user or machine disabled the
outboard write
MCHOFF
            BTST        #OUTC_W1,(A0)
            BEQ.S       MOREWR                    ; MACHINE DISABLED -TRY AGAIN
;
;                       Here the user has the write port disabled ---tell him
;
            MOVE.W      #IOEowdsbl,D7
            RTS
;
;                      Here start the hardware write operation if appropriate
;
MAYWRTS
            BSR         DISINTS
            LEA         WB_FLG2+1,A0
            BCLR        #EMPT_W2,(A0)             ;WAS IT EMPTY
            BEQ.S       MAYBYE                    ; NOPE NOTR APPROPRIATE TO STA
T HARDWARE-IT SHUD BE ON
            BSR         STRTXMIT                  ;ENABLE XMITT INTERRUPT PROCES
ING
MAYBYE
            BSR         ENBINTS
```

```
e buffer
;
WRTONE
            MOVE.B      (A3)+,D7
            SUBQ.W      #1,D2
            BSR.S       UPUTCHR                 ;PUT THE USER'S CHARACTER INTO T
HE WRITE BUFFER
            BCC.S       COMWR                   ;IF WRPROB THEN FOUND BY UPUTCHR
;
;
;                       Note that d7 is all set up to point at err if found by up
utchr
;                         thus a simple return is all thats required
WRPROB
            RTS
COMWEX
            BSR.S       MAYWRTS         ;MAY HAVE TO START A WRITE EVEN THO WRITE
BUFFER NOT FULL -RAN
                                        ;OUT OF USER CHARACTERS!!!!
            MOVE.W      #IOok,D7
            RTS
;
 ;(SP
;-------------------------------------------------------------------------
; UPUTCHR  --- User level put character routine, puts the character into the wr
te buffer.
;
;                       This routine assumes that somebody has already checked t
 see that
;                       there is enuf room in the buffer
;                       The character to be put is in d7.
;                       If there are any problems, then the horr5or code is set
nto D7
;                        and the carry bit is set
;                       If all ok then carry bit is cleared
UPUTCHR
            MOVEM.L     D1-D6/A0-A6,-(SP)
            BSR         DISINTS
            LEA         WB_FILLP,A1              ; POINTER TO ADDRESS IN A1
            MOVE.L      (A1),A0                 ;A0 NOW POINTS TO THE FILL POSI
ION OF WRITE BUFFER
            MOVE.B      D7,(A0)+                ;Coming into this routine -bunch
a checks made to ensure
                                                ;   at least one character is a
ailable. This then is the
                                                ;   "get" of that character.
            MOVE.L      A0,(A1)               ;BUMP THE FILL POINTER--ADJUST BELO
 IFF NECESSARY
;
;           Now check for various things such as pointing beyond end of physica
 buffer
;             hi and lo water marks , and consistency of buffer sizes.
;
            LEA         WRTBUF+WBFLEN,A2          ;PHYSICAL ADDRESS OF END OF BU
FER IN A2
            CMPA.L      A0,A2                   ;Note ao incremented above to p
int to next logical
                                                ; character position
                                                ; Compare is  physical end of b
ffer - next assumed
                                                ;    character address
            BGT.S       NOWRAP                  ; HERE  + OK --NO WRAP AROUND O
 ADDRESS
;
```

```
            LEA         WRTBUF,A0                   ;ADDRESS OF TOP OF WRITE BUFFE
            MOVE.L      A0,(A1)                     ; WB_FILLP NOW RESET
;
;                       From here on out we will be screwing with things that th
 interrupt routines
;                         may well fiddle with, hence disable interrupts and hop
 for the best
;
NOWRAP
            LEA         WB_FREE,A0
            SUBQ.W      #1,(A0)                     ;SINCE WE WROTE CHAR, ONE LESS
REE SPACE
            BEQ.S       FULLUP                      ;IF TO FULLUP THEN THE BUFFER I
 FULL
            BGT.S       WRWAT                       ; IF + THEN SEE WHAT WATER MARK
 ARE LIKE
;
;                       Here we got severe problems if number of free spaces is
 0
;
            BSR       ENBINTS                     ;REENABLE INTERRUPTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #IOEwszerr,D7               ;SIZING ERROR
            MOVE.W      #1,CCR                      ;SET ERROR
            RTS
;                                       HERE THE BUFFER IS FULL SET FLAGS
FULLUP
            LEA         WB_FLG2+1,A1
            BSET        #FULL_W2,(A1)                ; FULL FLAG
            LEA         WB_FLG1+1,A1
            ANDI.B        #HILOMSK,(A1)
            BSET        #BGHI_W1,(A1)                ;ABOVE HI WATER MARK
            BSR          ENBINTS                    ;REENABLE INTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #0,CCR                       ;FULL BUFFER IS NOT ERROR
            RTS

;
;                       Here check the hi/lo water marks etc set flags user may
omeday want
;
WRWAT
            BSR.S       WRTWAT
            BSR       ENBINTS
            MOVEM.L     (SP)+,D1-D6/A0-A6
            MOVE.W      #0,CCR                      ; ALL OK
            RTS
;-------------------------------------------------------------------------------
----
;  WRTWAT  -checks water marks for WRITING
;
;           NOTES: This routine assumes that interrupts are disabled prior
;                  to its being provoked.
WRTWAT

            MOVEM.L     A0-A2/D1/D2,-(SP)
            LEA         WB_FLG1+1,A0              ;SETR UP FLAG WORD FOR MANIPS
            ANDI.B        #HILOMSK,(A0)
            LEA         WB_SIZE,A1                  ;ACTUAL SIZE OF BUFFER
            LEA         WB_FREE,A2                  ; TAKE AWAY FREE SPACE, LEAVES
UMBR CHARS IN BUFFR
            MOVE.W      (A1),D1                     ; HOW DOES NMBR CHARS COMPARE T
 LOW WATER MARK CR-LOWA
            SUB.W       (A2),D1                     ; =0 THEN AT LOW WATER
```

```
            CMP.W       (A1),D1
            BEQ.S       ATLOW
            BLT.S       BELOLOW
;
;                       Here obviously + so compare with hi water marks- current
y do not deal
;                         with the absurdity of user setting hi<lo etc.
;
            LEA         WB_SIZE,A1                  ; actual buffer size to d2 (ev
ntually)
            MOVE.W      (A1),D2
            LEA         WB_HIWA,A1                  ; numbr of chars in hiwater ma
k
            SUB.W       (A1),D2
            CMP.W       D2,D1                       ;hiwater mark - actual nmbr ch
rs in byuffer
            BEQ.S       ATHIW
            BLT.S       MORHIW
            BRA         BYW

ATLOW
            BSET        #ATLO_W1,(A0)
            BRA         BYW
BELOLOW
            BSET        #LTLO_W1,(A0)
            BRA         BYW
ATHIW
            BSET        #ATHI_W1,(A0)
            BRA         BYW
MORHIW
            BSET        #BGHI_W1,(A0)
BYW
            MOVEM.L     (SP)+,A0-A2/D1/D2
            RTS
;-------------------------------------------------------------------------------
---
  ;(SP
;           NOTE:     it is assumed that these routines are protected from inter
upts
;
; STRTXMIT - start xmit interrupt process by enabling UART to interrupt
;            on transmit buffer empty.
; STOPXMIT - stop xmit interrupt process by disabling UART to interrupt
;            on transmit buffer empty.
;
STRTXMIT
,MOVEQ      #XMITENB,D1                 ;ENABLE XMIT INT
,BRA.S      SXTGETB
STOPXMIT
,MOVEQ      #XMITDIS,D1                 ;DISABLE XMIT INT
;
SXTGETB     BSR         GETBASE                     ;GET UART BASE ADDRESS
,MOVE.B     CMDREGI(A0),D0              ;GET CURRENT CMD REG
            ANDI.B      #CLRD3D2,D0                 ;CLEAR BITS D3 & D2
,OR.B       D1,D0                       ;DON'T CHANGE OTHER BITS
,MOVE.B     D0,CMDREGI(A0)              ;SAVE CHANGED CMD REG
,RTS
  ;(SP
;*******************************************************************************
*************************
;
; DCOMINT - DataCom Interrupt routine for XMIT/RCV interrupts.
;
;           NOTE: I f we find some way to use only 1 driver to play with 2 port
```

```
;                    CRITICAL: if an interrupt occurs, then both the receive buffe
 full and the xmit
;                              buffer empty could be true simultaneously, so we mu
t test both.
;                                  However, only once thru the test then rte
;                              Currently the priority is reads then writes
;                    The fact that we have an interrupt for read or write means th
t we are not user
;                           disabled, so we don't have to check that condition.
;
DCOMINT     MOVEM.L     D0-A6,-(SP)                 ;SAVE ALL REGISTERS
,BSR        GETBASE                       ;GET UART BASE ADDRESS
;
; If Receive interrupt then see if should process character.
;
REACHK
,MOVE.B     STATRI(A0),D0                 ;GET STATUS OF UART
            MOVE.B      D0,D7                       ; STORE STATUS FOR TEST XMIT I
TERRUPT LATER
            BTST        #S_RCVF,D0                  ;TEST FOR RECEIVE BUFFER FULL
,BNE.S      DCIRCVC                       ;PROCESS RECEIVED CHAR
;
; Not Receive, if Transmit interrupt then see if can send character
;       NOTE: THIS TESTS D7 WHICH ALLOWS US TO COME THRU HERE AFTER A READ CHE
K DONE
WRICHK
            BTST        #S_WRTE,D7                  ;XMIT BUFFER EMPTY?
,BEQ.S      DCIEXIT                       ;NO, UNKNOWN INTERRUPT - EXIT
DCIPX       BSR         PRXMIT                      ;YES, PROCESS XMIT
DCIEXIT     MOVEM.L     (SP)+,D0-A6                 ;EXIT-RESTORE REGISTERS
PTRRTE      RTE                                     ;EXIT INTERRUPT
;-------------------------------------------------------------------------------
; process received character
;
DCIRCVC     MOVE.B      DATAREG(A0),D0              ;GET CHAR/CLEARS INTERRUPT
,BCLR       #BITD7,D0                     ;CLEAR D7 OF CHAR JUST IN CASE
            LEA         BF_PROF,A1                  ; SEE IF ANY PROTOCOLS AT ALL-
CHECK HI BYTE
            BTST        #PROT_P2,(A1)
            BNE.S       PROTS                       ; IF SET THEN A PROTOCOL EXIST
 -FOOEY
;
;                                        HERE just put character into read buffer
and get out since
;                                        no protocols are required.
;
            BSR.S       MPUTBFR
            BRA.S       WRICHK
;-----------------------------------------
;MPUTBFR --PUT A CHARACTER INTO THE READ BUFFER AND RETURN -ADJUST COUNTERS/POI
TERS AS REQUIRED
;           COMING IN   D7- CONTAINS STATUS WORD  D0 CONTAINS CHARACTRER
;                       A0 POINTS TO UART
MPUTBFR
;
            LEA         RB_FILLP,A1                 ; POINTER TO FILL POINTER IN A
            LEA         RB_FREE,A2
            MOVE.W      (A2),D1                     ;MOVE THE FREE CHARACTER COUNT
TO D1
            TST.W       D1                          ;SEE IF ANY FREE SPACE IN BUFF
R
            BEQ.S       OVFLOW                      ; NO FREE SPACE, ADDING THIS C
ARACTER WUD OVERFLOW BUFFER
;
```

```
;
            MOVE.L      (A1),A3                     ; A3 NOW POINTS TO BUFFER
            MOVE.B      D0,(A3)+                    ; AUTO ADJUST POINTER
            MOVE.L      A3,(A1)                     ; RESET THE FILL POINTER RB_FIL
LP
;
;                                                   ;Here check to see if pointer wraps around
 end of buffer
;
            LEA         RDBUF+RBFLEN,A5
            CMPA.L      A3,A5                       ;END OF BUFF-CURRENT LOC
            BGT.S       INORRP                      ; IF  0 THENB NO READ WRAP
            LEA         RDBUF,A5
            MOVE.L      A5,(A1)                     ;ADJUST POINTER
;
INORRP
            SUBQ.W      #1,D1                       ; DECREMENTR THE FREE COUNT
            MOVE.W      D1,(A2)                     ; ADJUST THE FREE CHARACTER COU
NT
            LEA         RB_FLG2+1,A2                ;NO LONGER IS BUFFER EMPTY
            BCLR        #EMPT_R2,(A2)               ;RESET EMPTY FLAG ANYHOO
            BSR         REDWAT                      ; ADJUST WATER MARKS FOR THE RE
AD BUFFER
                                                    ; REDWATR SCRIBBLES OVER REGS
            TST.W       D1                          ;SEE IF REALLY FULL
            BEQ.S       FULLRD
            MOVE.W      #0,CCR
            RTS
FULLRD
            LEA         RB_FLG2+1,A1
            BSET        #FULL_R2,(A1)
            MOVE.W      #0,CCR
            RTS
;                                                   Here if the character were put in the buf
fer would overflow
;                                                   In actuality data would be overwritten bu
t buffer bounds would
;                                                   still be observed. If no protocols, then
can the data. If
;                                                   protocols then set the carry as an error
indicator.
OVFLOW
            LEA         RB_FLG2+1,A1
            BSET        #LOST_R2,(A1)     ; SET DATA LOST FLAG -THROW BYTE AWAY AN
D RETURN
            MOVE.W      #1,CCR
            RTS
;-------------------------------------------------------------
;*********************TEMP IGNORE ALL PROTOCOLS*********************************
*****
PROTS
            LEA         BF_PROF+1,A1                ;IF LINE TYPE OF HANDSHAKE
      *
            BTST        #LINE,(A1)                  ;BETWEEN PRINTER & DRIVER
       *
            BNE.S       DCIEXIT                     ;THEN IGNORE CHARACTER
         *
;
          *
; Here examine xon/xoff and enq/ack type protocols
;
            BRA.S       DCIEXIT                     ;EXIT - STOPPED XMIT
 *                    *
;
```

```
*****************************
  ;(SP
; PRXMIT - process transmission interrupt
;            Just send the next character if possible
;
;            ENTRY : (A0) = UART Base address
PRXMIT
            LEA         BF_PROF,A1                  ;FIRST SEE IF ANY PROTOCOLS EN
BLED AT ALL
            BTST        #PROT_P2,(A1)               ; IF SET THEN A PROTOCOL EXIST
            BNE         PROWRT
            BSR         MGETCHR                     ;ATTEMPT TO WRITE A CHAR FROM
UFFER
            RTS
;                                       NOTE: no hairy checking of protocols curr
ntly done cuz first test
;                                       is to run sans protocols.
;-------------------------------------------------------------------
;MGETCHR - GET A CHARACTR FROM BUFFER TO MACHINE PORT
;
MGETCHR
;                                        First see if hve to force a line feed ou
;           LEA         WB_FLG2+1,A1
            BTST        #AULF_W2,(A1)               ; IS AUTO FEED REQUIRED?
            BEQ.S       NOLF                        ; NO
;                       Here auto lf is required iff last char out was CR
;
            BCLR        #CRTF_W2,(A1)               ; SEE IF CR WAS LAST CHARACTER
OUT?
            BEQ.S       NOLF                        ;NOPE
;
;                       HERE ADD THE LINE FEED-DONT ADJUST POINTERS
;
            MOVE.B      #LF,DATAREG(A0)
            MOVE.W      #0,CCR
            RTS
;
;                       HERE -NORMAL CHAR FROM BUFFER PROCESSING
;
NOLF
            LEA         WB_EMPTY,A1                 ; POINTER TO EMPTY POINTER
            LEA         WB_FREE,A2                  ;POINTER TO # OF FREE SPACES I
 BUFFER
            LEA         WB_SIZE,A4                  ; MAX SIZE OF BUFFER POINTER
            MOVE.W      (A4),D1                     ; D1 HAS MAX BUF SIZE
            SUB.W       (A2),D1                     ;SIZE-FREE SPACE =NUMBER CHARS
IN BUFFER (D1)
            BEQ.S       EMPTYB                      ; EMPTY BUFFER
            BLT.S       UNDFLOW                     ; IF < 0 CHARACTERS THEN DATA
NDERFLOW
;
;                                       Here we assume there is a character to se
d
;
            MOVE.L      (A1),A3                     ;A3 NOW POINTS DIRECTLY AT CHA
ACTER TO MOVE OUT
            CMPI.B      #CR,(A3)                    ; SEE IF THIS IS A CARRIAGE RE
URN
            BNE.S       NOCR
            LEA         WB_FLG2+1,A5                ; SET CR FLAG  FOR USE IF AULF
FLAG SET
            BSET        #CRTF_W2,(A5)
NOCR
            MOVE.B      (A3)+,DATAREG(A0)           ; PUSH CHARACTER OUT
```

```
;
;                                       ;Here check to see if pointer wraps around
 end of buffer
;
            LEA         WRTBUF+WBFLEN,A5
            CMPA.L      A3,A5                       ;END OF BUFF-CURRENT LOC
            BGT.S       INOWRP                      ; IF + THEN  NO WRAP
            LEA         WRTBUF,A5
            MOVE.L      A5,(A1)                     ;ADJUST POINTER
;
INOWRP
;   ***********************************Temporary kludge coming up --the full flag
 should really be reset
;           on the second interrupt after xmit interrupts enabled, since first i
nterrupt just primes
;           this routine to shove characters out. the very first interrupt only
means the
;           interrupt mechanism is working, not that characters made it to anywh
ere buit the
;           data regiuster.
            LEA         WB_FLG2+1,A1
            BCLR        #FULL_W2,(A1)               ; OBVIUOSLY ALWAYS TRUE EXCEPTI
NG KLUDGE NOTE ABOVE
            ADDQ.W      #1,(A2)                     ; ONE MORE FREE CHARACTER
            SUBQ.W      #1,D1                       ; ONE LESS CHAR IN BUFFER
            BEQ.S       EMPTYB
            BLT.S       UNDFLOW                     ; IF < 0  U NOW HAVE AN EMPTY B
UFFER WHICH MEANS
                                                    ; THE NEXT INTERRUPT INDICATRES
 THAT THE CHAR IS GONE
                                                    ; AND IF NO MORE CHARS, THEN MU
ST SHUT DOWN XMIT
                                                    ; OF CHARS (TURN OF XMIT INT EN
ABLE)
            BSR         WRTWAT                      ;ADJUST WATER MARKS
            MOVE.W      #0,CCR                      ; ALL IS STILL OK
            RTS
;
EMPTYB                                      ; THE BUFFER IS EMPTY , RESET ALL POINTERS
, ETC. THIS IS THE
                                            ; NORMAL WAY FOR A WRITE  TO TERMINATE, IN
CLUDING A TRIP THRU
                                            ; THE UNDFLOW ROUTINE.
            LEA         WB_FILLP,A1                 ; BUFFER FILLUP POINTER POINTER
            LEA         WRTBUF,A2                   ; BUFFER ADDRESS
            MOVE.L      A2,(A1)+
            MOVE.L      A2,(A1)+                    ; RESET POINTERS
            MOVE.W      #WBFLEN,(A1)+
            MOVE.W      #WBFLEN,(A1)+               ;RESET THE SIZE PARAMETERS
            LEA         WB_FLG1,A1
            CLR.W       (A1)
            ADDQ.L      #1,A1                       ;POINT TO FLAGS
            BSET        #LTLO_W1,(A1)
            ADDQ.L      #2,A1                       ;POINT TO NEXT FLAG
            BSET        #EMPT_W2,(A1)               ; EVEN THO EMPTY DO NOT STOPXMI
T UNTIL LAST INTERRUPT
            MOVE.W      #0,CCR
            RTS
;
;                                           Now handle the underflo which if happens c
nce is normal
;                                           termination mode for writing
;
UNDFLOW
```

```
 FOR A WRITE)
            BSR         STOPXMIT                    ; SHUT DOWN WRITE PORT
            MOVE.W      #1,CCR                      ; signal any protocol of end o
 the line
            RTS
;-------------------------------------------------------------------------
;
PROWRT
;           LEA         BF_INTL+1,A1
;           BTST        #ENQFLG,(A1)                ;SEE IF SHOULD SEND A ENQ
;           BEQ.S       PXTNENQ                     ;NO
;           BCLR        #ENQFLG,(A1)                ;SEND THE ENQ CHAR
;           MOVE.B      #ENQ,DATAREG(A0)            ;CLEARS INTERRUPT
;
PXTEXIT     RTS
;
  ;(SP
;*************************************************************************
*************************
; DCTLINT - Data Com Control interrupt service routine.
;           Ignores the interrupt if wasn't a DataCom Control interrupt,
;           therefore an Apple slot interrupt, or if NOT Line type
;           handshake method.  Always clears the interrupt.
;
DCTLINT     MOVEM.L     D0-A6,-(SP)                 ;SAVE REGISTERS
,BSR.S      INITDCC                         ;CLEAR INTERRUPT/(D0) = PORT A
;                                           See if any protocols at all and if so if
ny are line prots
;
            LEA         BF_PROF,A1                  ;HI ORDER BYTE OF FLAG
            BTST        #PROT_P2,(A1)+              ;SET U7P NEXT BYTE OF FLAG IN
ASE WE MOVE THRU
            BNE.S       DCLEXIT                     ; NO PROTOCOLS--GET OUT
;
;  HERE WE HAVE PROTOCOLS
; If (type of handshake <> Line) then exit
;
,BTST       #LINE,(A1)
,BEQ.S      DCLEXIT                         ;NOT LINE HANDSHAKE, EXIT
;
; Determine which Line is used as Busy line Port A
;
            BSR.S       FINDLIN                     ;NEEDS A1 = PTR TO BF_PROF+1
;
;set or clear Busy depending on state of line and whether it's Busy inverted or
not
;
;           LEA         BF_INTL+1,A2
;           MOVE.B      (A2),D1                     ;SAVE BUSY FLAG
;           MOVE.W      #DISINT4,SR                 ;DISABLE INTS
;           BSET        #BUSY,(A2)                  ;ASSUME LINE IS BUSY = TRUE
;           BSR.S       TSTLINE                     ;TEST LINE & INVERTED FLAG
;           BNE.S       DCLEXIT                     ;IS BUSY
;           BCLR        #BUSY,(A2)
;
; if wasn't Busy before then start up transmission process
;
;           BTST        #BUSY,D1                    ;TEST SAVED BUSY STATE
;           BEQ.S       DCLEXIT                     ;WASN'T BUSY
;           BSR         STRTXMIT                    ;START XMIT IF BUFFER NOT EMPT
;
DCLEXIT     MOVEM.L     (SP)+,D0-A6                 ;EXIT-RESTORE REGISTERS
,RTE
  ;(SP
```

```
;           stop interrupt if unknown Apple slot device is interrupting.
;           ASSUMES :  DDR for Port A is untampered and set at $80
;           Exit : (D0) = Port A with IOX toggled
;                  (A0) = address of Port A
;
INITDCC     LEA         NHIRA.L,A0
,MOVE.B     (A0),D0                     ;READ PORTA W/O HANDSHAKE
,BCHG       #7,D0                       ;TOGGLE IOX
,MOVE.B     D0,(A0)                     ;WRITE OUT CHANGED IOX
,RTS
;
; CALLIDCC - Call INITDCC when driver unmounted and get a DataCom Control
;            interrupt.  Toggles IOX to clear level 1 interrupt.
;
CALLIDCC    MOVEM.L     D0/A0,-(SP)             ;SAVE REGS USED BY INITDCC
,BSR.S      INITDCC
,MOVEM.L    (SP)+,D0/A0                 ;RESTORE REGS
,RTE
  ;{$P
;
; FINDLIN - Find which Line is used for Handshaking in Port A
;           ENTRY : (A1) = address of PTRFLAGS+1
;           EXIT  : (D3) = Bit # in Port A specifying line used for Busy
;
FINDLIN     MOVEQ       #1,D3                   ;BIT NUMBER IN PORT A CORRESPON
DING TO
,MOVEQ      #CTSLIN,D4                  ;FLAG BIT NUMBER
;
; Assumes that it will always find a line flag set
;
FLNLOOK     BTST        D4,(A1)                 ;IS BIT SET?
,BNE.S      FLNGOT                      ;YES, D3 PORT A BIT FOR DC 0
,ADDQ.B     #2,D3
,ADDQ.B     #1,D4                       ;TRY NEXT BIT FLAG
,CMPI.B     #DCDLIN+1,D4                ;DID LAST FLAG
,BNE.S      FLNLOOK                     ;NO
;
; if (DataCom flag is set) then bit# := bit# + 1 - DC 1 bits in Port A are next
bit up
;
FLNGOT      BTST        #DATACOM,(A1)
,BEQ.S      FLNEXIT
,ADDQ.B     #1,D3
FLNEXIT     RTS
;
; TSTLINE - test Port A line used for Busy and the inverted flag to show if
;           Busy or NOT Busy.
;           ENTRY : (A1) = address of PTRFLAGS+1
;                   (D0) = Port A
;                   (D3) = bit number in Port A of Line used by Busy
;           EXIT  : (NE) = Busy
;                   (EQ) = NOT Busy
;
TSTLINE     BTST        D3,D0                   ;Create Line Boolean
,SNE        D4
,BTST       #INVBUSY,(A1)               ;Create Inverted Boolean
,SNE        D5
,EOR.B      D4,D5                       ;IF RESULT IS $FF THEN BUSY
,RTS

  ;{$P
;*************************************************************************
**********
;
```

```
;           variables.  Initialize UART from Printer Control Table.
;
COMCLR      BSR         DISINTS                     ;DISABLE INTERRUPTS
            BSR         INIWRBF                     ;INIT BUFFER & CONTROL VARIABLE
S
            BSR         INIRDBF                     ;INIT BUFFER & CONTROL VARIABLE
S
 BSR        SETUART                     ;INIT UART FROM CONSTANTS & TABLE
,BSR        ENBINTS                     ;ENABLE INTERRUPTS
,RTS
;*************************************************************************
******************
;
; COMBSY - UNITBUSY
;           PASCAL BOOLEAN TRUE RETURNED IN D0 IF THERE ARE ANY CHARACTERS IN RE
AD BUFFER
;
COMBSY
            LEA         RB_FLG2+1,A0
            BTST        #EMPT_R2,(A0)
            SEQ         D0                          ;IF BIT NOT SET THEN = 0; CHARA
CTRERS EXIST D0 =111111
            ANDI.B      #TRUE,D0                    ;CONVERT FROM BOOLEAN TO PASCAL
 BOOLEAN-
            RTS
  ;(*$P
;*************************************************************************
******************
;
; COMUNMT - UNITUNMOUNT
;           Turnoff interrupt capabilities of COMM  driver & current DataCom
;
COMUNMT     BSR         DISINTS                     ;DISABLE INTERRUPTS
,
,BSR        GETBASE                     ;GET UART BASE
,MOVE.B     #TURNOFF,CMDREGI(A0)        ;TURNOFF UART
;
; have vectors point to a RTe instruction
;
,LEA        CALLIDCC,A0                 ;HAVE DATA COM CONTROL INT
,MOVE.L     A0,VEC1.W                   ;RESET IOX TO CLEAR THE INT
,LEA        VEC4.W,A1                   ;ASSUME DC0 (VECTOR 4)
            LEA         BF_PROF+1,A2
,BTST       #DATACOM,(A2)               ;IS IT DC0
,BEQ.S      PUNUSE0                     ;YES
,LEA        VEC2.W,A1                   ;DC1 IS VECTOR 2
PUNUSE0     MOVE.L      A0,(A1)                     ;PTR TO RTE IN CURRENT DC VECTO
R
;
; Restore Interrupts
;
,BSR        ENBINTS
,RTS
  ;(*$P
;*************************************************************************
******************
;
; COMST - UNITSTATUS
  call the Table change or buffer free Functions
;
COMST       CMPI.W      #TBLSTATE,D2                ;VALID FUNCTION CODE
,BHI.S      PSTERR                      ;NO
,MOVE.W     (A3),D0                     ;GET PARAMETER
,LEA        PSTTBL,A1                   ;TURN THE FUNCTION CODE INTO
```

```
,JMP        0(A1,D2.W)                  ;DO FUNCTION
;
; Invalid Function Code Error
;
PSTERR      MOVE.W      #INVFNC,D7
,RTS
;
; THE COM DRIVER STATUS JUMP TABLE
;
PSTTBL      DATA.W      STWBUF-PSTTBL           ;WRITE BUFFER FREE SPACE
            DATA.W      STRBUF-PSTTBL           ;READ BUFFER FREE SPACE
            DATA.W      STBAUD-PSTTBL           ;SET BAUD RATE
,DATA.W     STPRITY-PSTTBL              ;SET PARITY
,DATA.W     STDTACOM-PSTTBL             ;SET DATA COM
,DATA.W     STWRDSZ-PSTTBL              ;SET WORD SIZE
,DATA.W     STHNDSK-PSTTBL              ;SET HANDSHAKE METHOD
            DATA.W      STWRHI-PSTTBL           ;SET WRITE BUFFER HI WATER MARK
            DATA.W      STWRLO-PSTTBL           ;SET WRITE BUFFER LOW WATER MAR
K
            DATA.W      STRDHI-PSTTBL           ;SET READ BUFFER HI WATER MARK
            DATA.W      STRDLO-PSTTBL           ;SET READ BUFFER LOW WATER MARK
            DATA.W      STRDSTS-PSTTBL          ;TELL READ STATUS
            DATA.W      STWTSTS-PSTTBL          ;TELL WRITE STATUS
            DATA.W      STALCTL-PSTTBL          ;TELL STATE ALL CONTROL BUFFERS
            DATA.W      STBFCTL-PSTTBL          ;TELL BUFFER CONTROL BUFFER
            DATA.W      STWTCTL-PSTTBL          ;TELL WRITE CONTROL  BUFFER
            DATA.W      STRDCTL-PSTTBL          ;TELL READ CONTROL BUFFER
            DATA.W      STOUTRD-PSTTBL          ;TURN OFF OUTBOARD READ
            DATA.W      STINRD-PSTTBL           ;TURN OFF INBOARD READ
            DATA.W      STOUTWT-PSTTBL          ;TURN OFF OUTBOARD WRITE
            DATA.W      STINWT-PSTTBL           ;TURN OFF INBOARD WRITE
            DATA.W      BWBCHR-PSTTBL           ;TELL #CHARS IN WRITE BUFFER
            DATA.W      BRBCHR-PSTTBL           ;TELL #CHARS IN READ BUFFER
;
  ;(SP
;
; STWBUF - Return to the user the Free space in the write buffer
;
STWBUF
            BSR         DISINTS                 ;DISABLE INTERRUPTS
            LEA         WB_FREE,A1
            MOVE.W      (A1),(A3)               ;WRITE BUFFER FREE SPACE
,BSR        ENBINTS                     ;ENABLE INTERRUPTS
,RTS
;
; STRBUF - Return to the user the Free space in the READ buffer
;
STRBUF
            BSR         DISINTS                 ;DISABLE INTERRUPTS
            LEA         RB_FREE,A1
            MOVE.W      (A1),(A3)               ;WRITE BUFFER FREE SPACE
            BSR         ENBINTS                 ;ENABLE INTERRUPTS
            RTS
;
; STBAUD - Set the Baud Rate
;
STBAUD
,CMPI.W     #MAXBAUD,D0                 ;IS IT A VALID PARAMETER
,BHI.S      SETERR                      ;NO
;
            LEA         BF_RDBD,A0                  ;WHERE TO PUT VALUE
,LEA        BAUDCNV,A1                  ;CONVERSION ARRAY
,BRA.S      SAVPARM                     ;SAVE CONVERTED PARAMETER
;
```

```
STPRITY
,CMPI.W     #MAXPRTY,D0                 ;IS IT A VALID PARAMETER
,BHI.S      SETERR                      ;NO
;
            LEA         BF_PART,A0                  ;WHERE TO PUT VALUE
,LEA        PRTYCNV,A1                  ;CONVERSION ARRAY
,BRA.S      SAVPARM                     ;SAVE CONVERTED PARAMETER
;
; STWRDSZ - Set the word size to transmit (7 or 8)
;
STWRDSZ
,CMPI.W     #MAXWRDS,D0                 ;IS IT A VALID PARAMETER
,BHI.S      SETERR                      ;NO
;
            LEA         BF_WRDS,A0                  ;WHERE TO PUT VALUE
,MOVE.B     D0,(A0)                     ;PUT IN WORD SIZE VALUE
,BRA.S      RSTUART                     ;RESET UART FROM TABLE
;
; common code to STBAUDR, STPRITY, STWRDSZ, STDTACOM, & STHNDSK
;
SAVPARM     MOVE.B      0(A1,D0.W),(A0)             ;SAVE CONVERTED PARAMETER
;
RSTUART     BSR         DISINTS                     ;DISABLE INTERRUPTS
RSTUART1    BSR         SETUART                     ;SETUP UART FROM TABLE
,BSR        ENBINTS                     ;ENABLE INTERRUPTS
,RTS
;
; Invalid Parameter error
;
SETERR      MOVE.W      #INVPRM,D7
,RTS
  ;(SP
;
; STDTACOM  - Change the DataCom being used.  Either DataCom 0 or 1.
;
STDTACOM
,CMPI.W     #MAXDTCM,D0                 ;IS IT A VALID PARAMETER
,BHI.S      SETERR                      ;NO
;
; change Table Flags
;
            LEA         BF_PROF+1,A0                ;FLAGS
,BSET       #DATACOM,(A0)               ;ASSUME PARAMETER=1
,TST.B      D0
,BNE.S      SDCCNGV                     ;IS DC 1, CHANGE VECTORS
,BCLR       #DATACOM,(A0)
;
; Change interrupt vectors.  The old vector point at a RTE instruction.
;
SDCCNGV     BSR         DISINTS                     ;DISABLE INTERRUPTS
,BSR        SETDCVEC                    ;SET VECTORS
,BRA.S      RSTUART1
;-------------------------------------------------------------------------------
---
;
; STHNDSK - Set Handshake type.  Convert parameter into the flags and put these
;           flag values into the Printer Control Table.  Don't need to reset
;           UART.
;
STHNDSK
,CMPI.W     #MAXHNDS,D0                 ;IS IT A VALID PARAMETER
,BHI.S      SETERR                      ;NO
;
,LEA        HNDSCNV,A1                  ;CONVERSION ARRAY
```

```
,MOVE.B     (A0),D2                 ;GET FLAG BYTE
,ANDI.B     #DCMFLGM,D2             ;REMOVE CURRENT HANDSHAKE FLAGS
,OR.B       D1,D2                   ;PUT IN NEW FLAGS
,MOVE.B     D2,(A0)                 ;RESTORE FLAGS
,RTS
  ;(SP
;
;STWRHI  -SET THE WRITE BUFFER HIGH WATER MARK
STWRHI
            LEA         WB_HIWA,A1
            MOVE.W      (A3),(A1)
            RTS
;
;
;STWRLO  -SET THE WRITE BUFFER LOW WATER MARK
STWRLO
            LEA         WB_LOWA,A1
            MOVE.W      (A3),(A1)
            RTS
;
;
;STRDHI  -SET THE READ BUFFER HIGH WATER MARK
STRDHI
            LEA         RB_HIWA,A1
            MOVE.W      (A3),(A1)
            RTS
;
;
;STRDLO  -SET THE READ BUFFER LOW WATER MARK
STRDLO
            LEA         RB_LOWA,A1
            MOVE.W      (A3),(A1)
            RTS
;
;
;
;STRDSTS -GET THE READ BUFFER STATUS
STRDSTS
            RTS
;
;
;
;STWTSTS -GET THE WRITE BUFFER STATUS
STWTSTS
            RTS
;
  ;(SP
;
;STALCTL -RETURN TO USER ALL CONTROL BUFFER VALUES
STALCTL
            BSR         STBFCTL
            BSR         STWTCTL
            BSR         STRDCTL
            RTS
;
;STBFCTL -RETURN TO USER ALL BUFFER CONTROL BUFFER VALUES
STBFCTL
            BSR         STTBLST
            RTS
;
;STWTCTL -RETURN TO USER ALL WRITE CONTROL BUFFER VALUES
STWTCTL
            RTS
;
```

```
            RTS
; STTBLST - Return to the user in the parameter block the state of the Buffer Co
ntrol Table.
;       ParameterBlock = record
;                           BaudRate  : integer; {range = 0..6}
;                           Parity    : integer; {range = 0..4}
;                           DataCom   : integer; {range = 0..1}
;                           WordSize  : integer; {range = 0..1}
;                           HandShake : integer; {range = 0..7}
;                         end;
;
STTBLST     CLR.L       D1                          ;MAKE SURE NO GARBAGE IN REGIST
ER
;
; GET BAUD RATE
;
            MOVE.W      #MAXBAUD,D0                 ;MAX BAUD RATE PARAMETER VALUE
            MOVE.B      BF_RDBD,D1                    ;CURRENT TABLE VALUE
            LEA         BAUDCNV,A0                  ;CONVERT TO INTEGER RANGE
            BSR.S       GETVAL
;
; GET PARITY
;
            MOVE.W      #MAXPRTY,D0                 ;MAX PARITY PARAMETER VALUE
            LEA         BF_PART,A0
            MOVE.B      (A0),D1                   ;CURRENT TABLE VALUE
            LEA         PRTYCNV,A0                    ;CONVERT TO INTEGER RANGE
            BSR.S       GETVAL
;
; GET DATACOM
;
            MOVE.B      BF_PROF+1,D1                ;GET FLAG BYTE WITH DATACOM FLA
G IN IT
            LSR.B       #7,D1                       ;TURN FLAG INTO A 0 OR A 1 INTE
GER
            MOVE.W      D1,(A3)+
;
; GET WORD SIZE
;
            MOVE.B      BF_WRDS,D1
            MOVE.W      D1,(A3)+
;
; GET HANDSHAKE
;
            MOVE.W      #MAXHNDS,D0                 ;MAX HANDSHAKE PARAMETER VALUE
            MOVE.B      BF_PROF+1,D1                ;CURRENT TABLE VALUE
            BCLR        #DATACOM,D1                 ;REMOVE DATACOM FLAG
            LEA         HNDSCNV,A0                  ;CONVERT TO INTEGER RANGE
;
; GET PARAMETER VALUE AN PUT IN PARAMETER BLOCK
;
GETVAL      CMP.B       0(A0,D0.W),D1               ;SEE WHICH CONVERSION VALUE = C
URRENT VALUE
            DBEQ        D0,GETVAL                   ;THE INDEX OF ONE = IS THE PARA
METER VALUE TO
            MOVE.W      D0,(A3)+                    ;RETURN TO USER IN PARAMETER BL
OCK
            RTS
 ;{$P
;
; STOUTRD -- STOP OUTBOARD READING
STOUTRD
            RTS
;
```

```
STINRD
            RTS
;
;
; STOUTWT -- STOP OUTBOARD WRITING
STOUTWT
            RTS
;
;
; STINWT -- STOP INBOARD WRITING
STINWT
            RTS
;
  ;(SP
;                                       FIND # CHARS IN WRITE BUFFER
BWBCHR
            LEA         WB_SIZE,A1
            LEA         WB_FREE,A2
            MOVE.W      (A1),D1                     ;SIZE IN D1
            SUB.W       (A2),D1                     ; SIZE -FREE =# CHARS
            MOVE.W      D1,D0                       ; IF 0 THEN 0 ELSE NUMB OF CHA
S
            RTS
;                                       FIND # CHARS IN READ BUFFGER
BRBCHR
            LEA         RB_SIZE,A1
            LEA         RB_FREE,A2
            MOVE.W      (A1),D1                     ;SIZE IN D1
            SUB.W       (A2),D1                     ; SIZE -FREE =# CHARS
            MOVE.W      D1,D0                       ; IF 0 THEN 0 ELSE NUMB OF CHA
S
            RTS
;(SP
;
; constant data area
;
; Conversion arrays for Set functions of Unitstatus
;
BAUDCNV     DATA.B      6,7,8,$A,$C,$E,$F           ;BAUD RATE
; 6=300,7=600,8=1200,A=2400,C=4800,E=9600,F=19200
;
PRTYCNV     DATA.B      0,1,3,5,7                   ;PARITY
; 0=DISABLED,1=ODD,3=EVEN,5=MARK XMIT/NO RCV,7=SPACE XMIT/NO RCV
;
HNDSCNV     DATA.B      $49                         ;LINE/CTS/INV
,DATA.B     $09                         ;LINE/CTS/NOT INV
,DATA.B     $51                         ;LINE/DSR/INV
,DATA.B     $11                         ;LINE/DSR/NOT INV
,DATA.B     $61                         ;LINE/DCD/INV
,DATA.B     $21                         ;LINE/DCD/NOT INV
,DATA.B     $02                         ;XON/XOFF
,DATA.B     $04                         ;ENQ/ACK
            DATA.B      $00                         ;NONE OF THE ABOVE PROTOCOLS
;=============================================================================
==============
; DEFAULT BUFFER  Control Table
;
;
,DATA.B     0                           ;FILL
DEFBWRT     DATA.B      $0E                         ;WRITE BAUD RATE-9600
DEFBRD      DATA.B      $0E                         ;READ BAUD RATE
DEFPART     DATA.B        0                         ;PARITY-DISABLED
DEFWRDS     DATA.B        0                         ;WORD SIZE = 8 BITS (1=7 BITS)
DEFINTRN    DATA.W      $1                          ;INTERNAL FLAG--SAVED ENTRY SR
```

```
                                        ;NOTE BOTH BYTES USED UPPER 9 M
EANS
                                        ;NO PROTOCOLS, FULL DUPLEX
 ;($P
;
;|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||
;
; Variable data area
;
; BUFFER CONTROL TABLE
;
BFRCTL
BF_WRBD     DATA.B      0                ;WRITE BAUD RATE
BF_RDBD     DATA.B      0                ;READ BAUD RATE
BF_PART     DATA.B      0                ;PARITY
BF_WRDS     DATA.B      0                ;WORD SIZE
BF_INTL     DATA.W      0                ;INTERNAL FLAGS
BF_PROF     DATA.W      0                ;PROTOCOL FLAGS-HANDSHAKE TYPE
& DATACOM
;
;===================================
;
;           WRITE BUFFER CONTROL TABLE
;
WRTCTL
WB_FILLP    DATA.L      0                ;BUFFER FILL POINTER
WB_EMPTY    DATA.L      0                ;BUFFER EMPTY POINTER
WB_SIZE     DATA.W      0                ;BUFFER SIZE
WB_FREE     DATA.W      0                ;AMOUNT OF BUFFER FREE SPACE
WB_HIWA     DATA.W      0                ;NUMBER OF BYTES IN HI WATER MA
RK
WB_LOWA     DATA.W      0                ;NUMBER OF BYTES IN LOW WATER M
ARK
WB_FLG1     DATA.W      0                ;FLAG WORD 1
WB_FLG2     DATA.W      0                ;FLAG WORD 2
;
;===================================
;
;           READ BUFFER CONTROL TABLE
;
RDCTL
RB_FILLP    DATA.L      0                ;BUFFER FILL POINTER
RB_EMPTY    DATA.L      0                ;BUFFER EMPTY POINTER
RB_SIZE     DATA.W      0                ;BUFFER SIZE
RB_FREE     DATA.W      0                ;AMOUNT OF BUFFER FREE SPACE
RB_HIWA     DATA.W      0                ;NUMBER OF BYTES IN HI WATER MA
RK
RB_LOWA     DATA.W      0                ;NUMBER OF BYTES IN LOW WATER M
ARK
RB_BENQ     DATA.W      0                ;NUMBER OF BYTES BETWEEN ENQ'S
RB_FLG1     DATA.W      0                ;FLAG WORD 1
RB_FLG2     DATA.W      0                ;FLAG WORD 2
;
; save areas for current SR
;
SAVESR1     DATA.W      0
;
;
; The Com Driver Read Buffer - 2k  bytes
;
RDBUF
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0      ;64
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0      ;128
```

```
RBFLEN      EQU         %-RDBUF                     ;READ BUFFER LENGTH
;
;
;-----------------------WRITE BUFFER-----------------------------------
WRTBUF
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         ;64
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         ;128
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         ;
            DATA.L      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0         ;256
WBFLEN      EQU         %-WRTBUF                    ;READ BUFFER LENGTH

            END         COMDRV
```