console

COLLABORATORS			
	TITLE : console		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

console

1.1 console.doc

```
 CD_ASKDEFAULTKEYMAP      CDInputHandler()      CloseDevice()
 CD_ASKKEYMAP             CMD_CLEAR             OpenDevice()
 CD_SETDEFAULTKEYMAP      CMD_READ              RawKeyConvert()
 CD_SETKEYMAP             CMD_WRITE
```

1.2 console.device/CD_ASKDEFAULTKEYMAP

```
NAME
    CD_ASKDEFAULTKEYMAP - get the current default keymap

FUNCTION
    Fill the io_Data buffer with the current console device
    default keymap, which is used to initialize console unit
    keymaps when opened, and by RawKeyConvert with a null
    keyMap parameter.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CD_ASKDEFAULTKEYMAP
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*keyMap)
    io_Data         struct KeyMap *keyMap
                    pointer to a structure that describes
                    the raw keycode to byte stream conversion.

RESULTS
    This function sets the io_Error field in the IOStdReq, and fills
    the structure pointed to by io_Data with the current device
    default key map.

BUGS

SEE ALSO
```

```
    exec/io.h, devices/keymap.h, devices/console.h
```

1.3 console.device/CD_ASKKEYMAP

```
NAME
    CD_ASKKEYMAP - get the current key map structure for this console

FUNCTION
    Fill the io_Data buffer with the current KeyMap structure in
    use by this console unit.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CD_ASKKEYMAP
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*keyMap)
    io_Data         struct KeyMap *keyMap
                    pointer to a structure that describes
                    the raw keycode to byte stream conversion.

RESULTS
    This function sets the io_Error field in the IOStdReq, and fills
    the structure the structure pointed to by io_Data with the current
     key map.

BUGS

SEE ALSO
    exec/io.h, devices/keymap.h, devices/console.h
```

1.4 console.device/CD_SETDEFAULTKEYMAP

```
NAME
    CD_SETDEFAULTKEYMAP - set the current default keymap

FUNCTION
    This console command copies the keyMap structure pointed to
    by io_Data to the console device default keymap, which is used
    to initialize console units when opened, and by RawKeyConvert
    with a null keyMap parameter.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CD_SETDEFAULTKEYMAP
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*keyMap)
    io_Data         struct KeyMap *keyMap
                    pointer to a structure that describes
```

```
                    the raw keycode to byte stream conversion.

RESULTS
    This function sets the io_Error field in the IOStdReq, and fills
    the current device default key map from the structure pointed to
    by io_Data.

BUGS

SEE ALSO
    exec/io.h, devices/keymap.h, devices/console.h
```

1.5 console.device/CD_SETKEYMAP

```
NAME
    CD_SETKEYMAP - set the current key map structure for this console

FUNCTION
    Set the current KeyMap structure used by this console unit to
    the structure pointed to by io_Data.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CD_SETKEYMAP
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*keyMap)
    io_Data         struct KeyMap *keyMap
                    pointer to a structure that describes
                    the raw keycode to byte stream conversion.

RESULTS
    This function sets the io_Error field in the IOStdReq, and fills
    the current key map from the structure pointed to by io_Data.

BUGS

SEE ALSO
    exec/io.h, devices/keymap.h, devices/console.h
```

1.6 console.device/CDInputHandler

```
NAME
    CDInputHandler - handle an input event for the console device

SYNOPSIS
    events = CDInputHandler(events, consoleDevice)
                    A0      A1

FUNCTION
    Accept input events from the producer, which is usually the
```

```
    rom input.task.

INPUTS
    events - a pointer to a list of input events.
    consoleDevice - a pointer to the library base address of the
        console device.  This has the same value as ConsoleDevice
        described below.

RESULTS
    events - a pointer to a list of input events not used by this
        handler.

NOTES
    This function is available for historical reasons.  It is
    preferred that input events be fed to the system via the
    WriteEvent command of the input.device.

    This function is different from standard device commands in
    that it is a function in the console device library vectors.
    In order to obtain a valid library base pointer for the
    console device (a.k.a. ConsoleDevice) call
    OpenDevice("console.device", -1, IOStdReq, 0),
    and then grab the io_Device pointer field out of the IOStdReq
    and use as ConsoleDevice.

BUGS

SEE ALSO
    input.device
```

1.7 console.device/CMD_CLEAR

```
NAME
    CMD_CLEAR - clear console input buffer

FUNCTION
    Remove from the input buffer any reports waiting to satisfy
    read requests.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CMD_CLEAR
    io_Flags        IOB_QUICK set if quick I/O is possible, else 0

BUGS

SEE ALSO
    exec/io.h, devices/console.h
```

1.8 console.device/CMD_READ

```
NAME
    CMD_READ - return the next input from the keyboard

FUNCTION
    Read the next input, generally from the keyboard.  The form of
    this input is as an ANSI byte stream: i.e. either ASCII text
    or control sequences.  Raw input events received by the
    console device can be selectively filtered via the aSRE and aRRE
    control sequences (see the write command).  Keys are converted
    via the keymap associated with the unit, which is modified
    with CD_AKSKEYMAP and CD_SETKEYMAP

    If, for example, raw keycodes had been enabled by writing
    <CSI>1{ to the console (where <CSI> is $9B or Esc[), keys
    would return raw keycode reports with the information from
    the input event itself, in the form:
    <CSI>1;0;<keycode>;<qualifiers>;0;0;<seconds>;<microseconds>q

    If there is no pending input, this command will not be
    satisfied, but if there is some input, but not as much as can
    fill io_Length, the request will be satisfied with the input
    currently available.

IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CMD_READ
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*buffer)
    io_Data         char buffer[]
                    a pointer to the destination for the characters to read
                    from the keyboard.

RESULTS
    This function sets the error field in the IOStdReq, and fills
        in the io_Data area with the next input, and io_Actual with
        the number of bytes read.

BUGS

SEE ALSO
    exec/io.h, devices/console.h
```

1.9 console.device/CMD_WRITE

```
NAME
    CMD_WRITE - write text to the display

FUNCTION
    Write a text record to the display.  Note that the RPort of
    the console window is in use while this write command is
    pending.
```

```
IO REQUEST
    io_Message      mn_ReplyPort set if quick I/O is not possible
    io_Device       preset by the call to OpenDevice
    io_Unit         preset by the call to OpenDevice
    io_Command      CMD_WRITE
    io_Flags        IOF_QUICK if quick I/O possible, else zero
    io_Length       sizeof(*buffer), or -1 if null terminated
    io_Data         char buffer[]
                    a pointer to a buffer containing the ANSI text
                    to write to the console device.

ANSI CODES SUPPORTED
    Independent Control Functions (no introducer) --
    Code    Name    Definition
    -----   ---     ----------------------------------------------
    00/ 7   BEL     BELL (actually a DisplayBeep)
    00/ 8   BS      BACKSPACE
    00/ 9   HT      HORIZONTAL TAB
    00/10   LF      LINE FEED
    00/11   VT      VERTICAL TAB
    00/12   FF      FORM FEED
    00/13   CR      CARRIAGE RETURN
    00/14   SO      SHIFT OUT
    00/15   SI      SHIFT IN
    01/11   ESC     ESCAPE

    Code or Esc Name Definition
    -----   --- ---- ------------------------------------------
    08/ 4   D   IND  INDEX: move the active position down one line
    08/ 5   E   NEL  NEXT LINE:
    08/ 8   H   HTS  HORIZONTAL TABULATION SET
    08/13   M   RI   REVERSE INDEX:
    09/11   [   CSI  CONTROL SEQUENCE INTRODUCER: see next list

    ISO Compatable Escape Sequences (introduced by Esc) --
    Esc   Name Definition
    ----- ---- ------------------------------------------------
    c     RIS  RESET TO INITIAL STATE

    Control Sequences, with the number of indicated parameters.
    i.e. <CSI><parameters><control sequence letter(s)>.  Note the
    last entries consist of a space and a letter.  CSI is either
    9B or Esc[.  A minus after the number of parameters (#p)
    indicates less is valid.  Parameters are seperated by
    semicolins, e.g. Esc[14;80H sets the cursor position to row
    14, column 80.
    CSI #p  Name Definition
    --- --- ---- ------------------------------------------------
    @   1-  ICH  INSERT CHARACTER
    A   1-  CUU  CURSOR UP
    B   1-  CUD  CURSOR DOWN
    C   1-  CUF  CURSOR FORWARD
    D   1-  CUB  CURSOR BACKWARD
    E   1-  CNL  CURSOR NEXT LINE
    F   1-  CPL  CURSOR PRECEEDING LINE
    H   2-  CUP  CURSOR POSITION
    I   1-  CHT  CURSOR HORIZONTAL TABULATION
```

```
    J   1-  ED   ERASE IN DISPLAY (only to end of display)
    K   1-  EL   ERASE IN LINE (only to end of line)
    L   1-  IL   INSERT LINE
    M   1-  DL   DELETE LINE
    P   1-  DCH  DELETE CHARACTER
    R   2   CPR  CURSOR POSITION REPORT (in Read stream only)
    S   1-  SU   SCROLL UP
    T   1-  SD   SCROLL DOWN
    W   n   CTC  CURSOR TABULATION CONTROL
    Z   1-  CBT  CURSOR BACKWARD TABULATION
    f   2-  HVP  HORIZONTAL AND VERTICAL POSITION
    g   1-  TBC  TABULATION CLEAR
    h   n   SM   SET MODE
    l   n   RM   RESET MODE
    m   n   SGR  SELECT GRAPHIC RENDITION
    n   1-  DSR  DEVICE STATUS REPORT
    t   1-  aSLPP SET PAGE LENGTH (private Amiga sequence)
    u   1-  aSLL SET LINE LENGTH (private Amiga sequence)
    x   1-  aSLO SET LEFT OFFSET (private Amiga sequence)
    y   1-  aSTO SET TOP OFFSET (private Amiga sequence)
    {   n   aSRE SET RAW EVENTS (private Amiga sequence)
    |   8   aIER INPUT EVENT REPORT (private Amiga Read sequence)
    }   n   aRRE RESET RAW EVENTS (private Amiga sequence)
    ~   1   aSKR SPECIAL KEY REPORT (private Amiga Read sequence)
     p  1-  aSCR SET CURSOR RENDITION (private Amiga sequence)
     q  0   aWSR WINDOW STATUS REQUEST (private Amiga sequence)
     r  4   aWBR WINDOW BOUNDS REPORT (private Amiga Read sequence)

    Modes, set with <CSI><mode-list>h, and cleared with
    <CSI><mode-list>l, where the mode-list is one or more of the
    following parameters, seperated by semicolins --
    Mode    Name Definition
    ------- ---- ------------------------------------------------
    20      LNM  LINEFEED NEWLINE MODE: if a linefeed is a newline
    >1      ASM  AUTO SCROLL MODE: if scroll at bottom of window
    ?7      AWM  AUTO WRAP MODE: if wrap at right edge of window

BUGS
    Does not display cursor in SuperBitMap layers.

SEE ALSO
    ROM Kernal Manual: libraries and devices, exec/io.h
```

1.10 console.device/CloseDevice

```
NAME
    Close -- close the console device

SYNOPSIS
    CloseDevice(IOStdReq)

FUNCTION
    This function closes software access to the console device,
    and informs the system that access to this device/unit which was
    previously opened has been concluded.  The device may perform
```

```
    certain house-cleaning  operations.  The I/O request structure
    is now free to be recycled.

INPUTS
    IOStdReq - pointer to an IOStdReq structure, set by OpenDevice

BUGS

SEE ALSO
    console.device/OpenDevice, exec/io.h
```

1.11 console.device/OpenDevice

```
NAME
    OpenDevice - a request to open a Console device

SYNOPSIS
    error = OpenDevice("console.device", unit, IOStdReq, 0 )
    D0                 A0              D0    A1        D1

FUNCTION
    The open routine grants access to a device.  There are two
    fields in the IOStdReq block that will be filled in: the
    io_Device field and possibly the io_Unit field.

    This open command differs from most other device open commands
    in that it requires some information to be supplied in the
    io_Data field of the IOStdReq block.  This initialization
    information supplies the window that is used by the console
    device for output.

    The unit number that is a standard parameter for an open call
    is used specially by this device.  A unit of -1 indicates that
    no actual console is to be opened, and is used to get a pointer
    to the device library vector ( which will be returned in the
    io_Device field of the IOStdReq block ).  A unit of zero binds
    the supplied window to a unique console.  Sharing a console
    must be done at a level higher than the device.  There are no
    other valid unit numbers.

IO REQUEST
    io_Data         struct Window *window
                    This is the window that will be used for this
                    console.  It must be supplied if the unit in
                    the OpenDevice call is 0 (see above).  The
                    RPort of this window is potentially in use by
                    the console whenever there is an outstanding
                    write command.

INPUTS
    "console.device" - a pointer to the name of the device to be opened.
    unit - the unit number to open on that device (0, or -1).
    IOStdReq - a pointer to a standard request block
    0 - a flag field of zero
```

```
RESULTS
    error - zero if successful, else an error is returned.

BUGS
    If a console.device is attached to a SUPERBITMAP window, the cursor
    will not be displayed.  In this case you are required to TURN OFF the
    console's cursor (with the standard escape sequence), and synthisize
    your own.  Memory loss and compatiblity problems are possible if the
    cursor is not turned off.

SEE ALSO
    console.device/CloseDevice, exec/io.h, intuition/intuition.h
```

1.12 console.device/RawKeyConvert

```
NAME
    RawKeyConvert - decode raw input classes

SYNOPSIS
    actual = RawKeyConvert(event, buffer, length, keyMap)
    D0                     A0     A1      D1      A2

    ConsoleDevice in A6 if called from Assembly Language.

FUNCTION
    This console function converts input events of type
    IECLASS_RAWKEY to ANSI bytes, based on the keyMap, and
    places the result into the buffer.

INPUTS
    event -  an InputEvent structure pointer.
    buffer - a byte buffer large enough to hold all anticipated
        characters generated by this conversion.
    length - maximum anticipation, i.e. the buffer size in bytes.
    keyMap - a KeyMap structure pointer, or null if the default
        console device key map is to be used.

RESULTS
    actual - the number of characters in the buffer, or -1 if
        a buffer overflow was about to occur.

ERRORS
    if actual is -1, a buffer overflow condition was detected.
    Not all of the characters in the buffer are valid.

NOTES
    This function is different from standard device commands in
    that it is a function in the console device library vectors.
    In order to obtain a valid library base pointer for the
    console device (a.k.a. ConsoleDevice) call
    OpenDevice("console.device", -1, IOStdReq, 0),
    and then grab the io_Device pointer field out of the IOStdReq
    and use as ConsoleDevice.

BUGS
```

```
SEE ALSO
    console.device/OpenDevice, exec/io.h,
        devices/inputevent.h, devices/keymap.h
```