

keyboard

COLLABORATORS

	<i>TITLE :</i> keyboard	
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>
WRITTEN BY		July 19, 2024

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	keyboard	1
1.1	keyboard.doc	1
1.2	keyboard.device/CMD_CLEAR	1
1.3	keyboard.device/KBD_ADDRESETHANDLER	1
1.4	keyboard.device/KBD_READEVENT	2
1.5	keyboard.device/KBD_READMATRIX	3
1.6	keyboard.device/KBD_REMRESETHANDLER	3
1.7	keyboard.device/KBD_RESETHANDLERDONE	4

Chapter 1

keyboard

1.1 keyboard.doc

```
CMD_CLEAR
KBD_ADDRESETHANDLER
KBD_READEVENT
KBD_READMATRIX
KBD_REMRESETHANDLER
KBD_RESETHANDLERDONE
```

1.2 keyboard.device/CMD_CLEAR

```
NAME
CMD_CLEAR -- Clear the keyboard input buffer.

FUNCTION
Remove from the input buffer any keys transitions 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_Command CMD_CLEAR
io_Flags IOB_QUICK set if quick I/O is possible
```

1.3 keyboard.device/KBD_ADDRESETHANDLER

```
NAME
KBD_ADDRESETHANDLER -- Add a keyboard reset handler.

FUNCTION
Add a function to the list of functions called to clean up
before a hard reset generated at the keyboard. The reset
handler is called as:
    ResetHandler(handlerData)
    a1
```

```

IO REQUEST
io_Message  mn_ReplyPort set
io_Device   preset by OpenDevice
io_Unit     preset by OpenDevice
io_Command  KBD_ADDRESETHANDLER
io_Data     a pointer to an interrupt structure.
            is_Data     the handlerData pointer described above
            is_Code     the Handler function address

```

NOTES

Few of the Amiga keyboard models generate the communication codes used to implement this reset processing. Specifically, only the Euro a1000 (rare), and the B2000 keyboard generate them.

The interrupt structure is kept by the keyboard device until a RemResetHandler command is satisfied for it, but the KBD_ADDRESETHANDLER command itself is replied immediately.

1.4 keyboard.device/KBD_READEVENT

NAME

KBD_READEVENT -- Return the next keyboard event.

FUNCTION

Read raw keyboard events from the keyboard and put them in the data area of the IORequest. If there are no pending keyboard events, this command will not be satisfied, but if there are some events, but not as many as can fill IO_LENGTH, the request will be satisfied with those 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_Command  KBD_READEVENT
io_Flags    IOB_QUICK set if quick I/O is possible
io_Length   the size of the io_Data area in bytes: there
            are sizeof(inputEvent) bytes per input event.
io_Data     a buffer area to fill with input events. The
            fields of the input event are:
            ie_NextEvent
            links the events returned
            ie_Class
            is IECLASS_RAWKEY
            ie_Code
            contains the next key up/down reports
            ie_Qualifier
            only the shift and numeric pad bits are set
            ie_SubClass, ie_X, ie_Y, ie_TimeStamp
            are not used, and set to zero

```

RESULTS

This function sets the error field in the IORequest, and fills the IORequest with the next keyboard events (but not partial

events).

1.5 keyboard.device/KBD_READMATRIX

NAME

KBD_READMATRIX -- Read the current keyboard key matrix.

FUNCTION

This function reads the up/down state of every key in the key matrix.

IO REQUEST INPUT

`io_Message` `mn_ReplyPort` set if quick I/O is not possible
`io_Device` preset by the call to `OpenDevice`
`io_Command` `KBD_READMATRIX`
`io_Flags` `IOB_QUICK` set if quick I/O is possible
`io_Length` the size of the `io_Data` area in bytes: this must be big enough to hold the key matrix.
`io_Data` a buffer area to fill with the key matrix:
an array of bytes whose component bits reflect each keys state: the state of the key for keycode `n` is at bit $(n \text{ MOD } 8)$ in byte $(n \text{ DIV } 8)$ of this matrix.

IO REQUEST OUTPUT

`io_Error`
`IOERR_BADLENGTH` - the `io_Length` was not exactly 13 bytes. The buffer is unchanged. This is only returned by V33/V34 kickstart.
`io_Actual` the number of bytes filled in `io_Data` with key matrix data, i.e. the minimum of the supplied length and the internal key matrix size.

NOTE

For V33/V34 Kickstart, `io_Length` must be set to exactly 13 bytes.

RESULTS

This function sets the error field in the `IORequest`, and sets matrix to the current key matrix.

1.6 keyboard.device/KBD_REMRESETHANDLER

NAME

KBD_REMRESETHANDLER -- Remove a keyboard reset handler.

FUNCTION

Remove a function previously added to the list of reset handler functions with `KBD_ADDRESETHANDLER`.

IO REQUEST

`io_Message` `mn_ReplyPort` set
`io_Device` preset by `OpenDevice`

```
io_Unit    preset by OpenDevice
io_Command KBD_REMRESETHANDLER
io_Data    a pointer to the handler interrupt structure.
```

1.7 keyboard.device/KBD_RESETHANDLERDONE

NAME

KBD_RESETHANDLERDONE -- Indicate that reset handling is done.

FUNCTION

Indicate that reset cleanup associated with the handler has completed. This command should be issued by all keyboard reset handlers so that the reset may proceed.

IO REQUEST

```
io_Message mn_ReplyPort set
io_Device  preset by OpenDevice
io_Unit    preset by OpenDevice
io_Command KBD_RESETHANDLERDONE
io_Data    a pointer to the handler interrupt structure.
```

NOTES

The keyboard processor itself performs the hardware reset, and will time out and perform the reset even if some reset handlers have not indicated yet that the reset may proceed. This timeout is several seconds.