NAME

usb_quirk - USB quirks module

SYNOPSIS

To compile this module into the kernel, place the following line in your kernel configuration file:

```
device usb
```

Alternatively, to load the module at boot time, place the following line in loader.conf(5):

```
usb_quirk_load="YES"
```

DESCRIPTION

The **usb_quirk** module provides support for dynamically adding and removing quirks for USB devices with usbconfig(8).

General quirks:

UQ_AUDIO_SWAP_LR
: swap left and right channels

UQ_AU_INP_ASYNC
: input is async despite claim of adaptive

UQ_AU_NO_FRAC
: do not adjust for fractional samples

UQ_AU_NO_XU
: audio device has broken extension unit

UQ_AU_VENDOR_CLASS
: audio device uses vendor class to identify itself

UQ_AU_SET_SPDIF_CM6206
: audio device needs special programming to enable S/PDIF audio output

UQ_BAD_ADC
: bad audio spec version number

UQ_BAD_AUDIO
: device claims audio class, but is not

UQ_BROKEN_BIDIR
printer has broken bidir mode

UQ_BUS_POWERED
device is bus powered, despite claim

UQ_HID_IGNORE
device should be ignored by hid class

UQ_KBD_IGNORE
device should be ignored by kbd class

UQ_KBD_BOOTPROTO
device should set the boot protocol

UQ_UMS_IGNORE
device should be ignored by ums class

UQ_MS_BAD_CLASS
does not identify properly

UQ_MS_LEADING_BYTE
mouse sends an unknown leading byte

UQ_MS_REVZ
mouse has Z-axis reversed

UQ_MS_VENDOR_BTN
mouse has buttons in vendor usage page

UQ_NO_STRINGS
string descriptors are broken

UQ_POWER_CLAIM
hub lies about power status

UQ_SPUR_BUT_UP
spurious mouse button up events

UQ_SWAP_UNICODE

has some Unicode strings swapped

UQ_CFG_INDEX_1
select configuration index 1 by default

UQ_CFG_INDEX_2
select configuration index 2 by default

UQ_CFG_INDEX_3
select configuration index 3 by default

UQ_CFG_INDEX_4
select configuration index 4 by default

UQ_CFG_INDEX_0
select configuration index 0 by default

UQ_ASSUME_CM_OVER_DATA
assume cm over data feature

UQ_IGNORE_CDC_CM
ignore cm descriptor

UQ_WMT_IGNORE
device should be ignored by wmt driver

USB Mass Storage quirks:

UQ_MSC_NO_TEST_UNIT_READY
send start/stop instead of TUR

UQ_MSC_NO_RS_CLEAR_UA
does not reset Unit Att.

UQ_MSC_NO_START_STOP
does not support start/stop

UQ_MSC_NO_GETMAXLUN
does not support get max LUN

UQ_MSC_NO_INQUIRY

fake generic inq response

UQ_MSC_NO_INQUIRY_EVPD
: does not support inq EVPD

UQ_MSC_NO_SYNC_CACHE
: does not support sync cache

UQ_MSC_SHUTTLE_INIT
: requires Shuttle init sequence

UQ_MSC_ALT_IFACE_1
: switch to alternate interface 1

UQ_MSC_FLOPPY_SPEED
: does floppy speeds (20kb/s)

UQ_MSC_IGNORE_RESIDUE
: gets residue wrong

UQ_MSC_WRONG_CSWSIG
: uses wrong CSW signature

UQ_MSC_RBC_PAD_TO_12
: pad RBC requests to 12 bytes

UQ_MSC_READ_CAP_OFFBY1
: reports sector count, not max sec.

UQ_MSC_FORCE_SHORT_INQ
: does not support full inq.

UQ_MSC_FORCE_WIRE_BBB
: force BBB wire protocol

UQ_MSC_FORCE_WIRE_CBI
: force CBI wire protocol

UQ_MSC_FORCE_WIRE_CBI_I
: force CBI with int. wire protocol

UQ_MSC_FORCE_PROTO_SCSI
force SCSI command protocol

UQ_MSC_FORCE_PROTO_ATAPI
force ATAPI command protocol

UQ_MSC_FORCE_PROTO_UFI
force UFI command protocol

UQ_MSC_FORCE_PROTO_RBC
force RBC command protocol

3G Datacard (u3g) quirks:

UQ_MSC_EJECT_HUAWEI
ejects after Huawei USB command

UQ_MSC_EJECT_SIERRA
ejects after Sierra USB command

UQ_MSC_EJECT_SCSIEJECT
ejects after SCSI eject command 0x1b0000000200

UQ_MSC_EJECT_REZERO
ejects after SCSI rezero command 0x010000000000

UQ_MSC_EJECT_ZTESTOR
ejects after ZTE SCSI command 0x850101011801010101010000

UQ_MSC_EJECT_CMOTECH
ejects after C-motech SCSI command 0xff52444556434847

UQ_MSC_EJECT_WAIT
wait for the device to eject

UQ_MSC_EJECT_SAEL_M460
ejects after Sael USB commands

UQ_MSC_EJECT_HUAWEISCSI
ejects after Huawei SCSI command 0x11060000000000000000000000000000

UQ_MSC_EJECT_TCT
    ejects after TCT SCSI command 0x06f504025270

UQ_MSC_DYMO_EJECT
    ejects after HID command 0x1b5a01

See */sys/dev/usb/quirk/usb_quirk.h* or run "usbconfig dump_quirk_names" for the complete list of supported quirks.

LOADER TUNABLE

The following tunable can be set at the loader(8) prompt before booting the kernel, or stored in loader.conf(5).

hw.usb.quirk.%d

The value is a string whose format is:

```
"VendorId ProductId LowRevision HighRevision UQ_QUIRK,..."
```

Installs the quirks **UQ_QUIRK,...** for all USB devices matching **VendorId** and **ProductId** which have a hardware revision between and including **LowRevision** and **HighRevision**.

VendorId, **ProductId**, **LowRevision** and **HighRevision** are all 16 bits numbers which can be decimal or hexadecimal based.

A maximum of 100 variables **hw.usb.quirk.0, .1, ..., .99** can be defined.

If a matching entry is found in the kernel's internal quirks table, it is replaced by the new definition.

Else a new entry is created given that the quirk table is not full.

The kernel iterates over the **hw.usb.quirk.N** variables starting at **N = 0** and stops at **N = 99** or the first non-existing one.

EXAMPLES

After attaching a **u3g** device which appears as a USB device on *ugen0.3*:

```
usbconfig -d ugen0.3 add_quirk UQ_MSC_EJECT_WAIT
```

Enable a Holtec/Keep Out F85 gaming keyboard on *ugen1.4*:

```
usbconfig -d ugen1.4 add_quirk UQ_KBD_BOOTPROTO
```

To install a quirk at boot time, place one or several lines like the following in loader.conf(5):

```
hw.usb.quirk.0="0x04d9 0xfa50 0 0xffff UQ_KBD_IGNORE"
```

SEE ALSO

usbconfig(8)

HISTORY

The **usb_quirk** module appeared in FreeBSD 8.0, and was written by Hans Petter Selasky <*hselasky@FreeBSD.org*>. This manual page was written by Nick Hibma <*n_hibma@FreeBSD.org*>.