Revised: 2004-Oct-21

This is the documentation of (hopefully) all possible error codes (and their interpretation) that can be returned from usbcore.

Some of them are returned by the Host Controller Drivers (HCDs), which device drivers only see through usbcore. As a rule, all the HCDs should behave the same except for transfer speed dependent behaviors and the way certain faults are reported.

****
* Error codes returned by usb_submit_urb *
****

Non-USB-specific:

0	URB submission went fine
-ENOMEM	no memory for allocation of internal structures

USB-specific:

-ENODEV	specified USB-device or bus doesn't exist
-ENOENT	specified interface or endpoint does not exist or is not enabled
-ENXIO	host controller driver does not support queuing of this type of urb. (treat as a host controller bug.)
-EINVAL	a) Invalid transfer type specified (or not supported) b) Invalid or unsupported periodic transfer interval c) ISO: attempted to change transfer interval d) ISO: number_of_packets is < 0 e) various other cases
-EAGAIN	a) specified ISO start frame too early b) (using ISO-ASAP) too much scheduled for the future wait some time and try again.
-EFBIG	Host controller driver can't schedule that many ISO frames.
-EPIPE	The pipe type specified in the URB doesn't match the endpoint's actual type.
-EMSGSIZE	(a) endpoint maxpacket size is zero; it is not usable in the current interface altsetting. (b) ISO packet is larger than the endpoint maxpacket. (c) requested data transfer length is invalid: negative or too large for the host controller.
-ENOSPC	This request would overcommit the usb bandwidth reserved for periodic transfers (interrupt, isochronous).
-ESHUTDOWN	The device or host controller has been disabled due to some

problem that could not be worked around.

-EPERM Submission failed because urb->reject was set.

-EHOSTUNREACH URB was rejected because the device is suspended.

-ENOEXEC A control URB doesn't contain a Setup packet.

****
* Error codes returned by in urb->status *
* or in iso_frame_desc[n].status (for ISO) *
****

USB device drivers may only test urb status values in completion handlers. This is because otherwise there would be a race between HCDs updating these values on one CPU, and device drivers testing them on another CPU.

A transfer's actual_length may be positive even when an error has been reported. That's because transfers often involve several packets, so that one or more packets could finish before an error stops further endpoint I/O.

0	Transfer completed successfully
-ENOENT	URB was synchronously unlinked by usb_unlink_urb
-EINPROGRESS	URB still pending, no results yet (That is, if drivers see this it's a bug.)
-EPROTO (*, **)	a) bitstuff error b) no response packet received within the prescribed bus turn-around time c) unknown USB error
-EILSEQ (*, **)	a) CRC mismatch b) no response packet received within the prescribed bus turn-around time c) unknown USB error Note that often the controller hardware does not distinguish among cases a), b), and c), so a driver cannot tell whether there was a protocol error, a failure to respond (often caused by device disconnect), or some other fault.
-ETIME (**)	No response packet received within the prescribed bus turn-around time. This error may instead be reported as -EPROTO or -EILSEQ.
-ETIMEDOUT	Synchronous USB message functions use this code to indicate timeout expired before the transfer completed, and no other error was reported by HC.
-EPIPE (**)	Endpoint stalled. For non-control endpoints, reset this status with usb_clear_halt().

-ECOMM	During an IN transfer, the host controller received data from an endpoint faster than it could be written to system memory
-ENOSR	During an OUT transfer, the host controller could not retrieve data from system memory fast enough to keep up with the USB data rate
-EOVERFLOW (*)	The amount of data returned by the endpoint was greater than either the max packet size of the endpoint or the remaining buffer size. "Babble".
-EREMOTEIO	The data read from the endpoint did not fill the specified buffer, and URB_SHORT_NOT_OK was set in urb->transfer_flags.
-ENODEV	Device was removed. Often preceded by a burst of other errors, since the hub driver doesn't detect device removal events immediately.
-EXDEV	ISO transfer only partially completed look at individual frame status for details
-EINVAL	ISO madness, if this happens: Log off and go home
-ECONNRESET	URB was asynchronously unlinked by usb_unlink_urb
-ESHUTDOWN	The device or host controller has been disabled due to some problem that could not be worked around, such as a physical disconnect.

(*) Error codes like -EPROTO, -EILSEQ and -EOVERFLOW normally indicate hardware problems such as bad devices (including firmware) or cables.

(**) This is also one of several codes that different kinds of host controller use to indicate a transfer has failed because of device disconnect. In the interval before the hub driver starts disconnect processing, devices may receive such fault reports for every request.

****
* Error codes returned by usbcore-functions *
* (expect also other submit and transfer status codes) *
****

usb_register():
-EINVAL error during registering new driver

usb_get_*/usb_set_*():
usb_control_msg():
usb_bulk_msg():
-ETIMEDOUT Timeout expired before the transfer completed.