Next Up Previous Contents Index

Next: Preface **Up:** USB Developer Pages **Contents** **Index**

Programming Guide for Linux USB Device Drivers

(c) 2000 by Detlef Fliegl

http://usb.cs.tum.edu

`$Id: usbdoc.tex,v 1.32 2000/12/25 18:36:26 deti Exp $`

This document can be found on http://usb.cs.tum.edu/usbdoc and can be downloaded from http://usb.cs.tum.edu/download/usbdoc

---

Detlef Fliegl
2001-01-08

Preface

The development of the Linux USB subsystem started in 1997 and in the meantime it was redesigned many times. This implied various changes of its internal structure and its API too. So it is even hard for experienced device driver developers to keep up to date with all ongoing discussions and current changes.

This document should give detailed information about the current state of the USB subsystem and its API for USB device drivers. The first section will deal with the basics of USB devices. You will learn about different types of devices and their properties. Going into detail you will see how USB devices communicate on the bus. The second section gives an overview of the Linux USB subsystem [2] and the device driver framework. Then the API and its data structures will be explained step by step. The last section of this document contains a reference of all API calls and their return codes.

Detlef Fliegl
2001-01-08

Institut für Informatik TU-München

Lehr- und Forschungseinheit Informatik X

Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechner

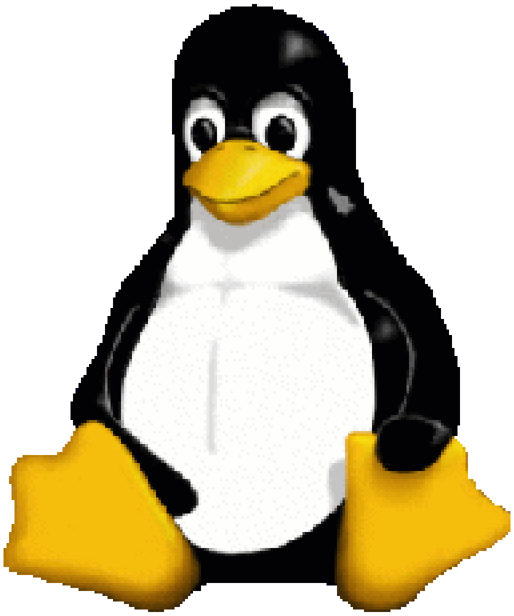
Prof. Dr. A. Bode

Universal Serial Bus Development for Linux

G. Acher & D. Fliegl & T. Sailer & R. Weissgärber

http://usb.cs.tum.edu

for



- Programming Guide for Linux USB Device Drivers
- People
- Related Sites / Links
- Download

USB-Team, $Date: 2000/03/14 22:29:01 $

Next Up Previous Index

Next: List of Figures **Up:** Programming Guide for Linux **Previous:** Preface **Index**

Contents

Detlef Fliegl
2001-01-08

Next Up Previous Contents

Next: About this document ... **Up:** Programming Guide for Linux **Previous:** Bibliography **Contents**

Index

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: The Universal Serial Bus **Up:** Programming Guide for Linux **Previous:** Contents **Contents** **Index**

List of Figures

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Host Controllers **Up:** Programming Guide for Linux **Previous:** List of Figures **Contents** **Index**

The Universal Serial Bus

In 1994 an alliance of four industrial partners (Compaq, Intel, Microsoft and NEC) started to specify the Universal Serial Bus (USB). The bus was originally designed with these intentions:

- Connection of the PC to the telephone
- Ease-of-use
- Port expansion

The specification (version 1.0) was first released in january 1996 and the latest official version 1.1 was released in september 1998 [4]. The document is still under development and a version 2.0 was announced in 1999. More information and all specification papers can be found in [1]. The USB is strictly hierarchical and it is controlled by one host. The host uses a master / slave protocol to communicate with attached USB devices. This means that every kind of communication is initiated by the host and devices cannot establish any direct connection to other devices. This seems to be a drawback in comparison to other bus architectures but it is not because the USB was designed as a compromise of costs and performance. The master / slave protocol solves implicitly problems like collision avoidance or distributed bus arbitration. The current implementation of the USB allows 127 devices to be connected at the same time and the communication bandwidth is limited to 12Mbit/s.

Subsections

Next Up Previous Contents Index

Next: Host Controllers **Up:** Programming Guide for Linux **Previous:** List of Figures **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: USB Devices and Transfer **Up:** The Universal Serial Bus **Previous:** The Universal Serial Bus
Contents **Index**

Host Controllers

Today the USB host controller is integrated on most motherboard chipsets. Older boards which are not equipped with such a controller can be upgraded by PCI cards with such host controllers. All these controllers are compatible with either the Open Host Controller Interface (OHCI by Compaq, Microsoft and National Semiconductor) or the Universal Host Controller Interface (UHCI by Intel [7]) standard. Both types have the same capabilities and USB devices do not have to care about the host controller. Basically the hardware of UHCI is simpler and therefore it needs a more complex device driver, which could cause slightly more CPU load.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Hubs **Up:** The Universal Serial Bus **Previous:** Host Controllers **Contents** **Index**

USB Devices and Transfer Characteristics

There are different types of USB devices as they can be used for different purposes. First a device can be self powered, bus powered or both. The USB can provide a power supply up to 500mA for its devices. If there are only bus powered devices on the bus the maximum power dissipation could be exceeded and therefore self powered devices exist. They need to have their own power supply. Devices that support both power types can switch to self powered mode when attaching an external power supply.

Even the maximum communication speed can differ for particular USB devices. The USB specification decides between low speed and full speed devices. Low speed devices (such as mice, keyboards, joysticks etc.) communicate at 1.5MBit/s and have only limited capabilities. Full speed devices (such as audio and video systems) can use up to 90% of the 12Mbit/s which is about 10Mbit/s including the protocol overhead.

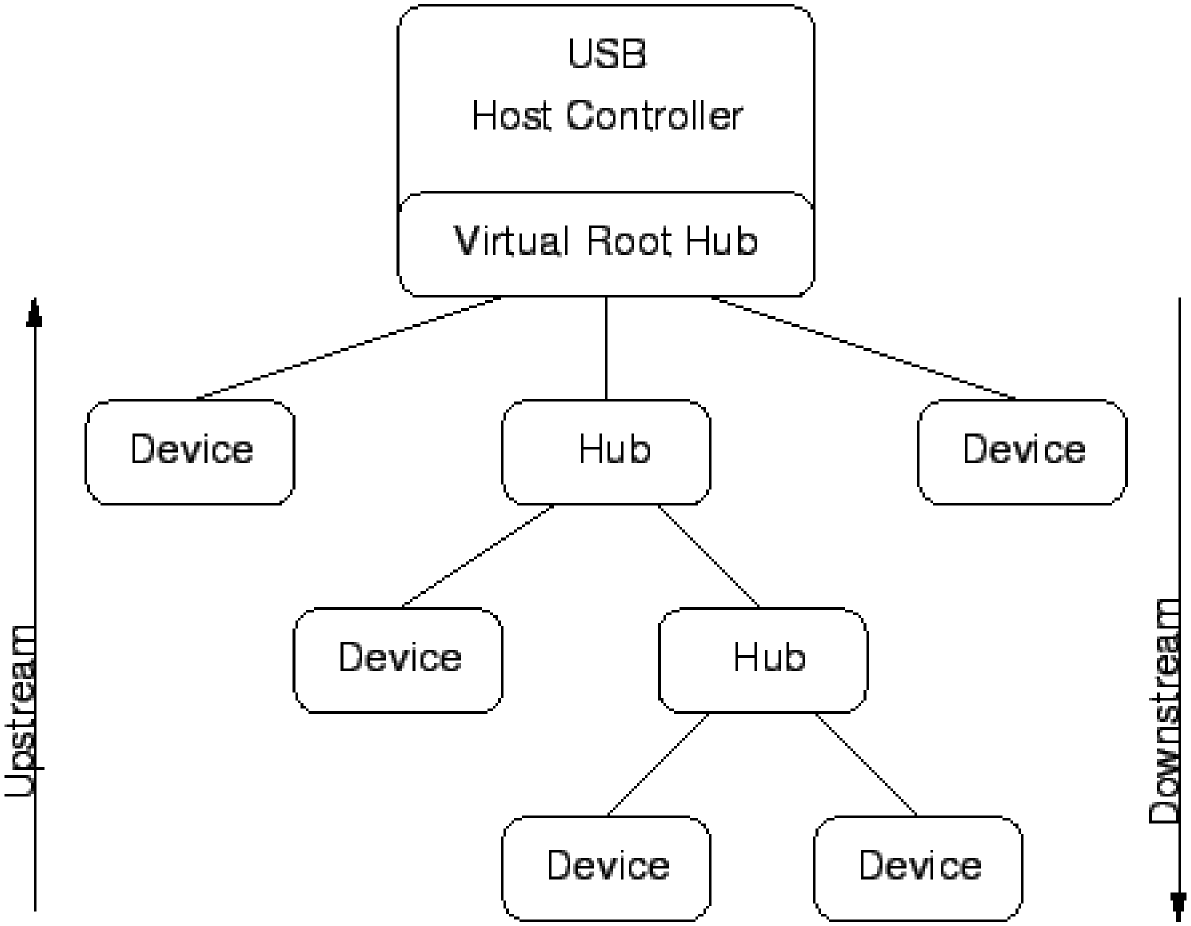


Figure 1:USB Topology

Subsections

Next Up Previous Contents Index

Next: Hubs **Up:** The Universal Serial Bus **Previous:** Host Controllers **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Data Flow Types **Up:** USB Devices and Transfer **Previous:** USB Devices and Transfer **Contents** **Index**

Hubs

Physically there exist a number of USB ports at the rear panel of a computer. These ports can be used to attach normal devices or a hub. A hub is a USB device which extends the number of ports (i.e. 2-8) to connect other USB devices. The maximum number of attachable devices is reduced by the number of hubs on the bus. Hubs are self- and/or bus powered full speed devices.

Normally the physical ports of the host controller are handled by a virtual root hub. This hub is simulated by the host controller's device driver and helps to unify the bus topology. So every port can be handled in the same way by the USB subsystem's hub driver (see figure 1).

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Enumeration and Device Descriptors **Up:** USB Devices and Transfer **Previous:** Hubs **Contents** **Index**

Data Flow Types

The communication on the USB is done in two directions and uses 3 different transfer types. Data directed from the host to a device is called downstream or OUT transfer. The other direction is called upstream or IN transfer. Depending on the device type different transfer variants are used:

- **Control transfers** are used to request and send reliable short data packets. It is used to configure devices and every one is required to support a minimum set of control commands. Here is a list of standard commands:
  - GET_STATUS
  - CLEAR_FEATURE
  - SET_FEATURE
  - SET_ADDRESS
  - GET_DESCRIPTOR
  - SET_DESCRIPTOR
  - GET_CONFIGURATION
  - SET_CONFIGURATION
  - GET_INTERFACE
  - SET_INTERFACE
  - SYNCH_FRAME

  Further control commands can be used to transfer vendor specific data.

- **Bulk transfers** are used to request or send reliable data packets up to the full bus bandwidth. Devices like scanners or scsi adapters use this transfer type.

- **Interrupt transfers** are similar to bulk transfers which are polled periodically. If an interrupt transfer was submitted the host controller driver will automatically repeat this request in a specified interval (1ms - 255ms).

- **Isochronous transfers** send or receive data streams in realtime with guaranteed bus bandwidth but without any reliability. In general these transfer types are used for audio and video devices.

Next Up Previous Contents Index

Next: Enumeration and Device Descriptors **Up:** USB Devices and Transfer **Previous:** Hubs **Contents** **Index**

Detlef Fliegl
2001-01-08

Enumeration and Device Descriptors

Whenever a USB device is attached to the bus it will be enumerated by the USB subsystem - i.e an unique device number (1-127) is assigned and then the device descriptor is read. Such a desciptor is a data structure which contains information about the device and its properties. The USB standard defines a hierarchy of descriptors (see figure 2).

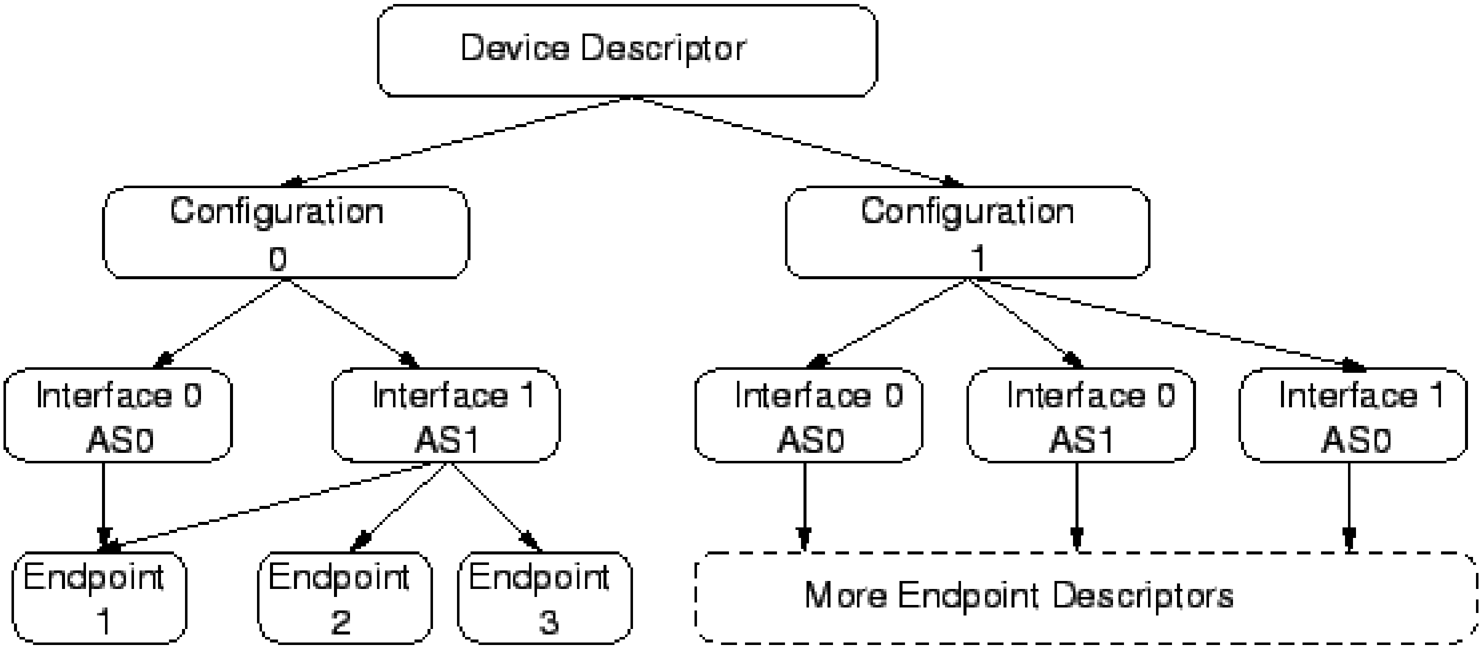


Figure 2:USB Descriptor Hierarchy

Subsections

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Device Classes **Up:** Enumeration and Device Descriptors **Previous:** Enumeration and Device Descriptors **Contents** **Index**

Standard Descriptors

- A **Device Descriptor** describes general information about a USB device. It includes information that applies globally to the device and all of the device's configurations. A USB device has only one device descriptor.
- The **Configuration Descriptor** gives information about a specific device configuration. A USB device has one or more configuration descriptors. Each configuration has one or more interfaces and each interface has zero or more endpoints. An endpoint is not shared among interfaces within a single configuration unless the endpoint is used by alternate settings of the same interface. Endpoints may be shared among interfaces that are part of different configurations without this restriction. Configurations can be activated exclusively by the standard control transfer `set_configuration`. Different configurations can be used to change global device settings like power consumption.
- An **Interface Descriptor** describes a specific interface within a configuration. A configuration provides one or more interfaces, each with zero or more endpoint descriptors describing a unique set of endpoints within the configuration. An interface may include alternate settings that allow the endpoints and/or their characteristics to be varied after the device has been configured. The default setting for an interface is always alternate setting zero. Alternate settings can be selected exclusively by the standard control transfer `set_interface`. For example a multifunctional device like a video camera with internal microphone could have three alternate settings to change the bandwidth allocation on the bus.

    1. Camera activated
    2. Microphone activated
    3. Camera and microphone activated

- An **Endpoint Descriptor** contains information required by the host to determine the bandwidth requirements of each endpoint. An endpoint represents a logical data source or sink of a USB device. Endpoint zero is used for all standard control transfers and there is never a descriptor for this endpoint. The USB specification [4] uses the term pipe for an endpoint too.

- **String Descriptors** are optional and provide additional information in human readable Unicode format. They can be used for vendor and device names or serial numbers.

Next Up Previous Contents Index

Next: Device Classes **Up:** Enumeration and Device Descriptors **Previous:** Enumeration and Device Descriptors   **Contents**   **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Human Interface Devices (HID) **Up:** Enumeration and Device Descriptors **Previous:** Standard Descriptors **Contents** **Index**

Device Classes

The standard device and interface descriptors contain fields that are related to classification: class, sub-class and protocol. These fields may be used by a host system to associate a device or interface to a driver, depending on how they are specified by the class specification [5]. Valid values for the class fields of the device and interface descriptors are defined by the USB Device Working Group (see also Figure 1).

Table 1:USB Device Classes

Device Class	Example Device
Display	Monitor
Communication	Modem
Audio	Speakers
Mass storage	Hard drive
Human interface	Data glove

Grouping devices or interfaces together in classes and then specifying the characteristics in a Class Specification allows the development of host software which can manage multiple implementations based on that class. Such host software adapts its operation to a specific device or interface using descriptive information presented by the device. A class specification serves as a framework defining the minimum operation of all devices or interfaces which identify themselves as members of the class.

Next Up Previous Contents Index
Next: Human Interface Devices (HID) **Up:** Enumeration and Device Descriptors **Previous:** Standard Descriptors **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: USB Device Drivers **Up:** Enumeration and Device Descriptors **Previous:** Device Classes

Contents **Index**

Human Interface Devices (HID)

The HID class [6] consists primarily of devices that are used by humans to control the operation of computer systems. Typical examples of HID class devices include:

- Keyboards and pointing devices for example, standard mouse devices, trackballs, and joysticks.
- Front-panel controls for example: knobs, switches, buttons, and sliders.
- Controls that might be found on devices such as telephones, VCR remote controls, games or simulation devices for example: data gloves, throttles, steering wheels, and rudder pedals.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: The Linux USB Subsystem **Up:** The Universal Serial Bus **Previous:** Human Interface Devices (HID) **Contents** **Index**

USB Device Drivers

Finding device drivers for USB devices presents some interesting situations. In some cases the whole USB device is handled by a single device driver. In other cases, each interface of the device has a separate device driver.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: The USB Device Driver **Up:** Programming Guide for Linux **Previous:** USB Device Drivers

Contents **Index**

The Linux USB Subsystem

In Linux there exists a subsystem called ``The USB Core'' with a specific API to support USB devices and host controllers. Its purpose is to abstract all hardware or device dependent parts by defining a set of data structures, macros and functions.

The USB core contains routines common to all USB device drivers and host controller drivers. These functions can be grouped into an upper and a lower API layer. As shown in figure 3 there exists an API for USB device drivers and another one for host controllers. The following section concentrates on the USB device driver layer, because the development for host controller drivers is already finished.

This section will give an overview of the USB framework by explaining entry points and the usage of API functions. If you are not familar with linux device drivers the following section might not be very useful. Appropriate literature can be found here [8], [9].

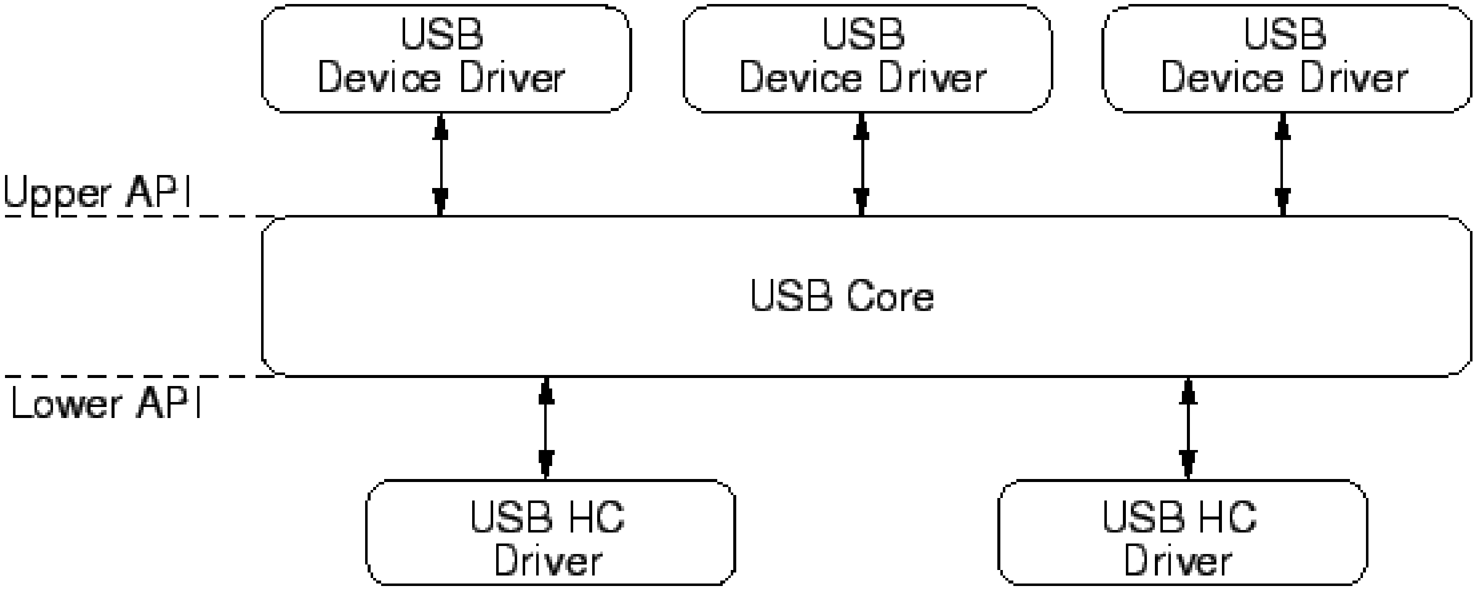


Figure 3:USB Core API Layers

Subsections

Next Up Previous Contents Index
Next: The USB Device Driver **Up:** Programming Guide for Linux **Previous:** USB Device Drivers
Contents **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Framework Data Structures **Up:** The Linux USB Subsystem **Previous:** The Linux USB Subsystem
Contents **Index**

The USB Device Driver Framework

USB devices drivers are registered and deregistered at the subsystem. A driver must register 2 entry points and its name. For specific USB devices (which are not suitable to be registered at any other subsystem) a driver may register a couple of file operations and a minor number. In this case the specified minor number and the 15 following numbers are assigned to the driver. This makes it possible to serve up to 16 similar USB devices by one driver. The major number of all USB devices is 180.

Subsections

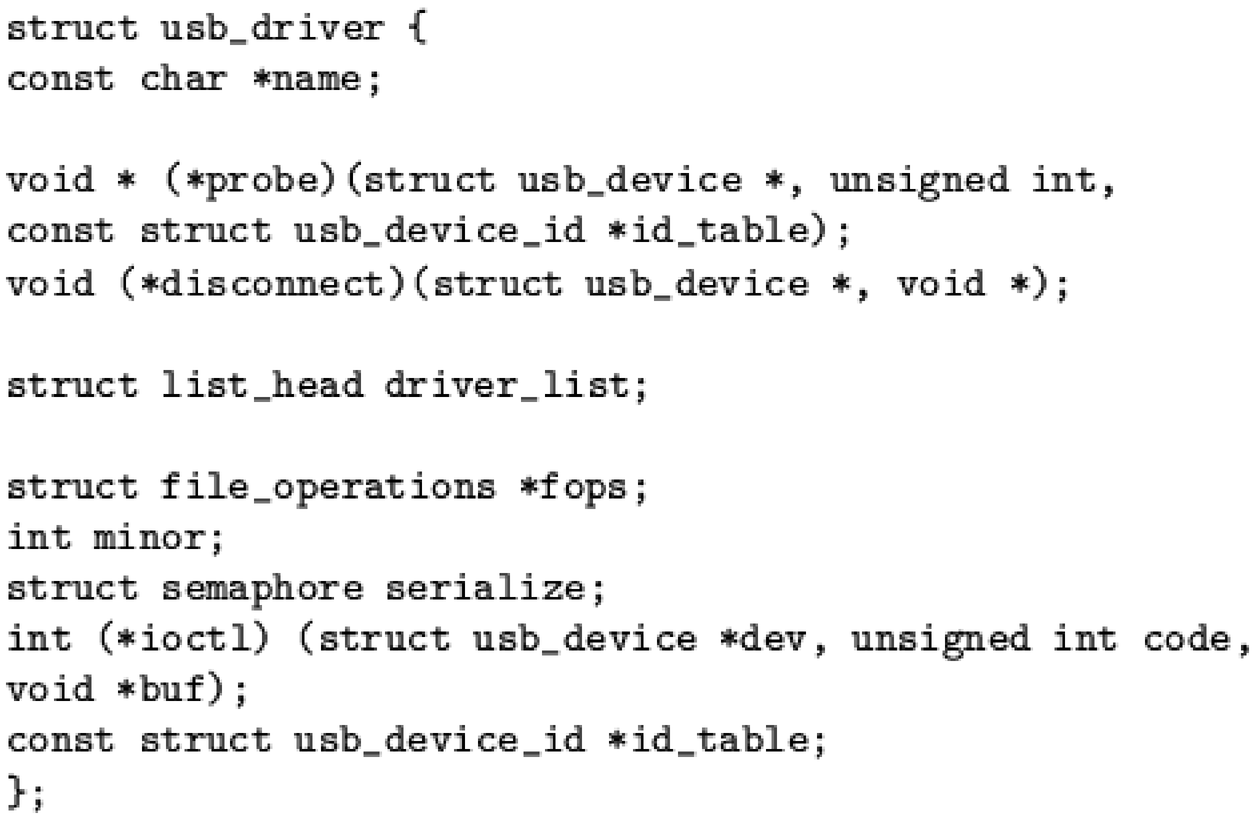
Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Framework Entry Points **Up:** The USB Device Driver **Previous:** The USB Device Driver
Contents **Index**

Framework Data Structures

All USB related functions or data structures follow the same naming convention and start with `usb_`. Figure 4 shows the structure needed to register a USB device driver at the subsystem.



```
struct usb_driver {
const char *name;

void * (*probe)(struct usb_device *, unsigned int,
const struct usb_device_id *id_table);
void (*disconnect)(struct usb_device *, void *);

struct list_head driver_list;

struct file_operations *fops;
int minor;
struct semaphore serialize;
int (*ioctl) (struct usb_device *dev, unsigned int code,
void *buf);
const struct usb_device_id *id_table;
};
```

Figure 4:usb_driver structure

- `name`: Usually the name of the module.
- `probe`: The entry point of the probe function.
- `disconnect`: The entry point of the disconnect function.
- `driver_list`: For internal use of the subsystem - initialize to {NULL,NULL}
- `fops`: The usual list of file operations for a driver
- `minor`: The base minor number assigned to this device (the value has to be a multiple of 16)
- `serialize`:
- `ioctl`:
- `id_table`:

Detlef Fliegl
2001-01-08

Framework Entry Points

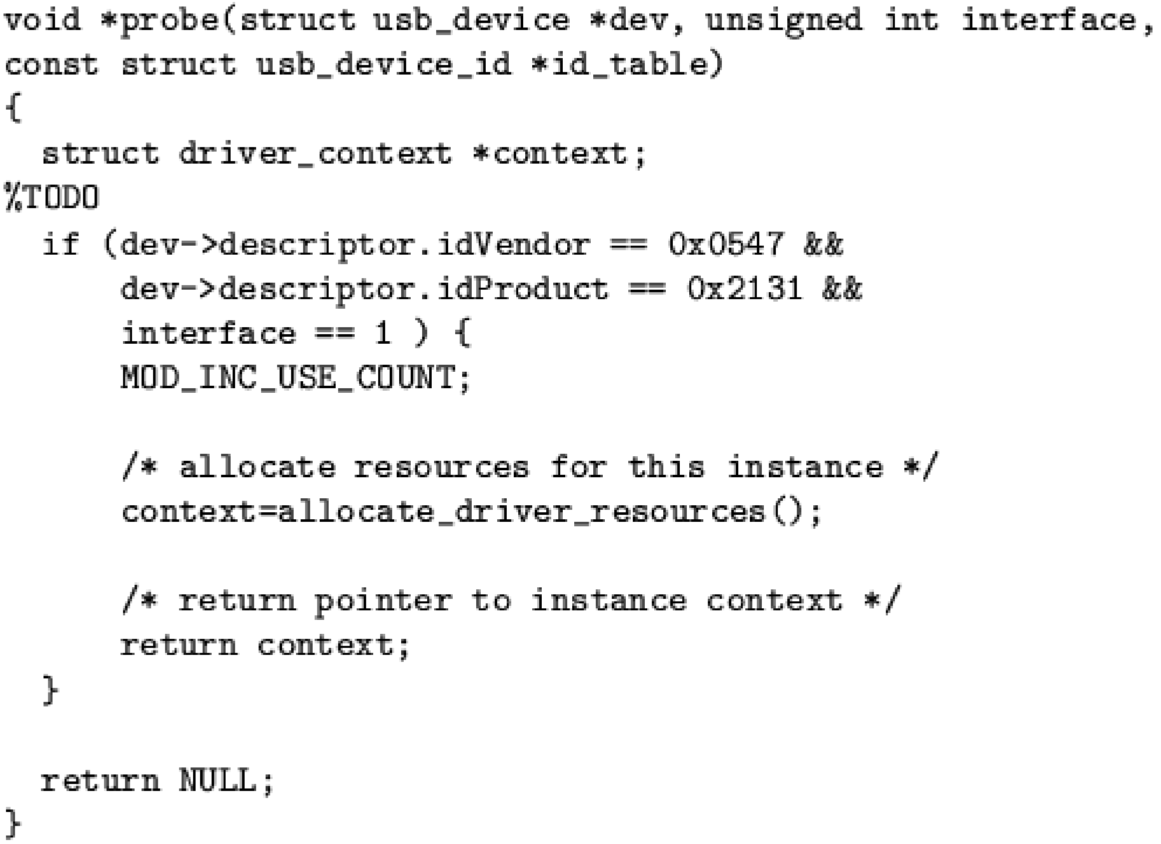
The USB driver framework adds two entry points to normal device drivers:

- `void *probe(struct usb_device *dev, unsigned int interface, const struct usb_device_id *id_table);` This entry point is called whenever a new device is attached to the bus. Then the device driver has to create a new instance of its internal data structures for the new device.

  The `dev` argument specifies the device context, which contains pointers to all USB descriptors. The `interface` argument specifies the interface number. If a USB driver wants to bind itself to a particular device and interface it has to return a pointer. This pointer normally references the device driver's context structure.

  Probing normally is done by checking the vendor and product identifications or the class and subclass definitions. If they match the interface number is compared with the ones supported by the driver. When probing is done class based it might be necessary to parse some more USB descriptors because the device properties can differ in a wide range.

  A simple probe routine is shown in figure 5.



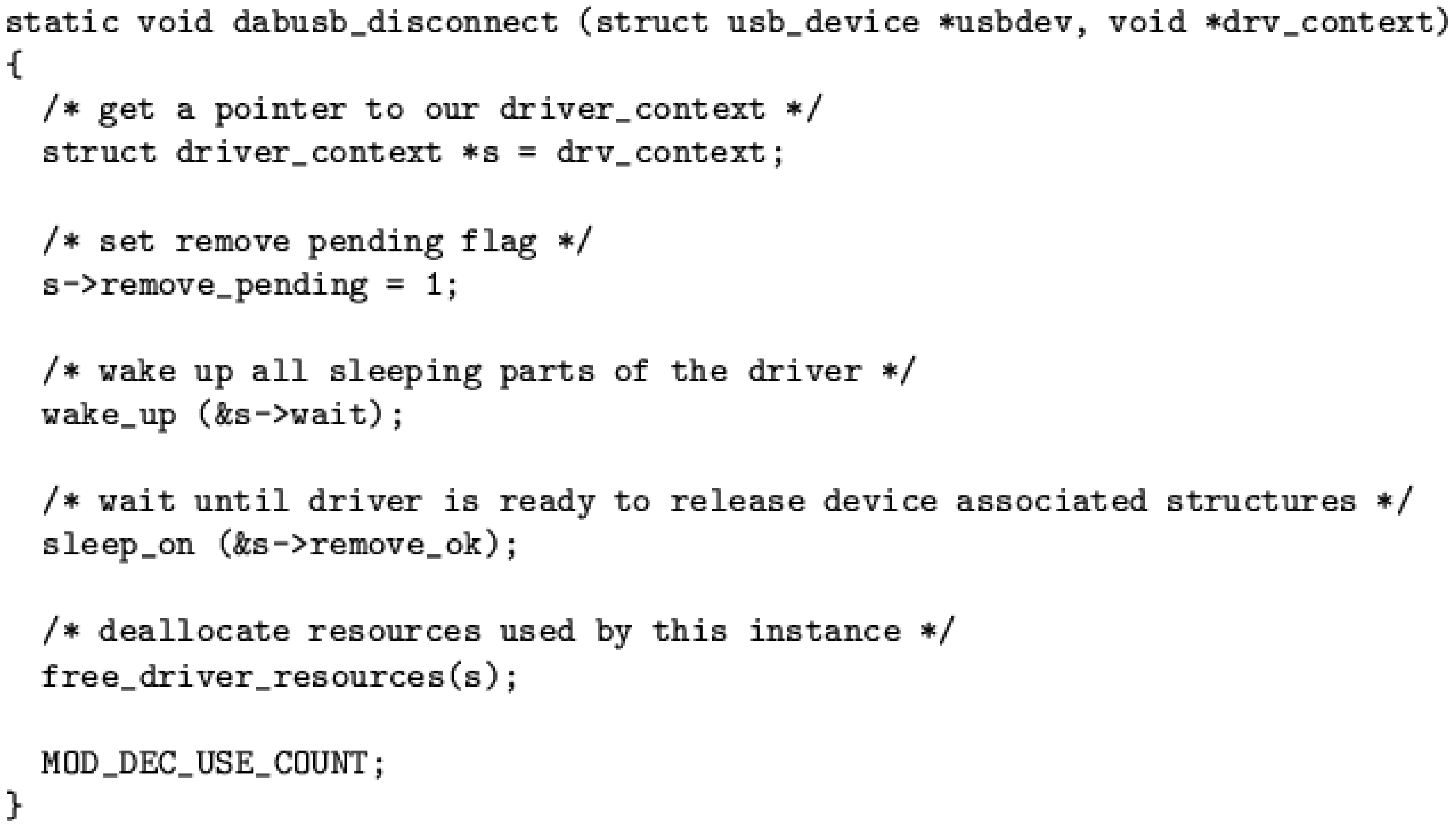
```
void *probe(struct usb_device *dev, unsigned int interface,
const struct usb_device_id *id_table)
{
  struct driver_context *context;
%TODO
  if (dev->descriptor.idVendor == 0x0547 &&
      dev->descriptor.idProduct == 0x2131 &&
      interface == 1 ) {
      MOD_INC_USE_COUNT;

      /* allocate resources for this instance */
      context=allocate_driver_resources();

      /* return pointer to instance context */
      return context;
  }

  return NULL;
}
```

Figure 5:A simple probe function

- `void disconnect(struct usb_device *dev, void *drv_context);` This function is called whenever a device which was served by this driver is disconnected.

  The argument `dev` specifies the device context and the `driver_context` returns a pointer to the previously registered `driver_context` of the probe function. After returning from the disconnect function the USB framework completly deallocates all data structures associated with this device. So especially the `usb_device` structure must not be used any longer by the usb driver.

  A simple disconnect function is shown in figure 6.



```
static void dabusb_disconnect (struct usb_device *usbdev, void *drv_context)
{
  /* get a pointer to our driver_context */
  struct driver_context *s = drv_context;

  /* set remove pending flag */
  s->remove_pending = 1;

  /* wake up all sleeping parts of the driver */
  wake_up (&s->wait);

  /* wait until driver is ready to release device associated structures */
  sleep_on (&s->remove_ok);

  /* deallocate resources used by this instance */
  free_driver_resources(s);

  MOD_DEC_USE_COUNT;
}
```

Figure 6:A simple disconnect function

Next Up Previous Contents Index

Next: Framework Functions **Up:** The USB Device Driver **Previous:** Framework Data Structures **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Configuring USB Devices **Up:** The USB Device Driver **Previous:** Framework Entry Points
Contents **Index**

Framework Functions

- `int usb_register(struct usb_driver *drv);`

  This function is used to register a new USB device driver at the subsystem. The argument `drv` points to a completely initialized `usb_driver` (see figure 4) structure. On success 0 is returned otherwise an error value is returned.

- `void usb_deregister(struct usb_driver *drv);`

  This function deregisters a formerly registerd USB device driver at the subsystem.

- `void usb_driver_claim_interface(struct usb_driver *driver, struct usb_interface *iface, void *drv_context);`

  This function is intended to be used by USB device drivers that need to claim more than one interface on a device at once when probing. The argument `driver` points to a completely initialized `usb_driver` structure. The `iface` argument points to a usb_interface structure which is part of the `usb_config_descriptor` which is accesible from the `usb_device structure` (given in the `probe` function). The `drv_context` pointer normally references the device driver's context structure (see return value of the probe function).

- `int usb_interface_claimed(struct usb_interface *iface);`

  This function is used to check if another device driver already has claimed the specified interface. The return value is 0 if the interface was not claimed by any driver.

- `void usb_driver_release_interface(struct usb_driver *driver, struct usb_interface *iface);`

  If a driver wants to release a previously claimed interface it has to call this function. In the `disconnect` function you do not have to release any interfaces that were additionally claimed in the `probe` function.

- `const struct usb_device_id *usb_match_id( struct usb_device *dev,`

```
        struct usb_interface *interface, const struct usb_device_id *id);
```

Next Up Previous Contents Index

Next: Configuring USB Devices **Up:** The USB Device Driver **Previous:** Framework Entry Points

Contents **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Descriptor Data Structures **Up:** The Linux USB Subsystem **Previous:** Framework Functions
Contents **Index**

Configuring USB Devices

The API includes a set of functions to select or query descriptors, configurations and alternate settings of devices. All these standard operations are done via control transfers to the device.

Subsections

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Standard Device Requests **Up:** Configuring USB Devices **Previous:** Configuring USB Devices
Contents **Index**

Descriptor Data Structures

The Linux USB subsystem describes the hierarchical structure of descriptors by extending or embedding the standard USB descriptors with or in a subsystem specific structure. This structure helps storing pointers to the selected configuration and interfaces.

The elements of these structures are only explained in detail as far as they are necessary for subsequent API calls. Detailed information about the descriptors can be found in `usb.h` and [4] section 9.5.

```
struct usb_device{
  ...
  struct usb_config_descriptor *actconfig;/* the active configuration */
  ...
  struct usb_device_descriptor descriptor;/* Descriptor */
  struct usb_config_descriptor *config; /* All of the configs */
}
```

The `usb_device` structure is the root of all USB specific descriptors. Sometimes it is necessary to parse the descriptors within a driver to configure the device or to setup transfer requests properly.

- Accessing all available configuration descriptors can be done like this:

  ```
  for (i = 0; i < dev->descriptor.bNumConfigurations; i++) {
    struct usb_config_descriptor *cfg = &dev->config[i];
    ...
  }
  ```

- Accessing all available interface descriptors of a particular configuration is done like this:

  ```
  for (j = 0; j < cfg->bNumInterfaces; j++) {
    struct usb_interface *ifp = &cfg->interface[j];
    ...
  }
  ```

To start the parsing of the active configuration simply use the `dev->actconfig` pointer.

- Accessing all alternate settings of a particular interface can be done like this:

```
for (k = 0; k < ifp->num_altsetting; k++) {
  struct usb_interface_descriptor *as = &ifp->altsetting[k];
  ...
}
```

  The active alternate setting can be accessed via
  `*as = &ifp->altsetting[ifp->act_altsetting]`

- Accessing all endpoint descriptors of a particular alternate setting can done like this:

```
for(l = 0; l < as->bNumEndpoints; l++) {
  struct usb_endpoint_descriptor *ep=&as->endpoint[k];
  ...
}
```

Next Up Previous Contents Index
Next: Standard Device Requests **Up:** Configuring USB Devices **Previous:** Configuring USB Devices
Contents **Index**

Detlef Fliegl
2001-01-08

Standard Device Requests

To query or set a particular configuration or alternate setting there exist a number functions. These commonly used functions setup standard device requests (control transfers for a specified device:

- `int usb_set_configuration(struct usb_device *dev, int configuration);`

  To activate a particular configuration use this function.
  The argument is of
  `0 <= configuration < dev->descriptor.bNumConfigurations`.
  Configuration 0 is selected by default after the device is attached to the bus.

- `int usb_set_interface(struct usb_device *dev, int interface, int alternate);`

  This function activates an alternate setting of a specified interface. The argument `interface` is of
  `0 <= interface < dev->actconfig->bNumInterfaces`.
  The argument `alternate` is of
  `0 <= alternate < dev->actconfig->interface[interface].num_altsetting`

- `int usb_get_device_descriptor(struct usb_device *dev);`

  This function rereads the complete descriptor tree from a particular device. It is called automatically whenever a device is attached to the bus or it may be called whenever a USB descriptor has changed.

- `int usb_get_descriptor(struct usb_device *dev, unsigned char desctype, unsigned char descindex, void *buf, int size);`

  Single USB descriptors can be read as raw data from a device. This function can be used to parse extended or vendor specific descriptors. The arguments `desctype` and `descindex` are documented in [4] section 9.4.3 and 9.5.

- `int usb_get_string(struct usb_device *dev, unsigned short langid, unsigned char index, void *buf, int size);`

  If a device, configuration or interface descriptor references a string index value (see [4] section 9.6.5) this function can be used to retrieve the string descriptor. According to the specification USB strings are coded as Unicode. If successful the function returns 0 otherwise an error code is returned.

- `int usb_string(struct usb_device *dev, int index, char *buf, size_t size);`

  This function simplifies `usb_get_string` by converting Unicode strings into ASCII strings.

- `int usb_get_status(struct usb_device *dev, int type, int target, void *data);`

  This USB control request is documented in [4] section 9.4.5.

- `int usb_clear_halt(struct usb_device *dev, int pipe);`

  If an endpoint is stalled (see [4] chapter 8.4.4) call this function to clear the STALL condition. STALL indicates that a function is unable to transmit or receive data, or that a control pipe request is not supported. The argument `endpoint` defines a pipe handle.

- `int usb_get_protocol(struct usb_device *dev, int ifnum);`

  This HID USB control request is documented in [6] section 7.2.5.

- `int usb_set_protocol(struct usb_device *dev, int protocol, int ifnum);` This HID USB control request is documented in [6] section 7.2.6.

- `int usb_get_report(struct usb_device *dev, unsigned char type, unsigned char id, int ifnum, void *buf, int size);`

  This HID USB control request is documented in [6] section 7.2.1

- `int usb_set_idle(struct usb_device *dev, int ifnum, int duration, int report_id);`

  This HID USB control request is documented in [6] section 7.2.4

Next Up Previous Contents Index

Next: USB Transfers **Up:** Configuring USB Devices **Previous:** Descriptor Data Structures **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Transfer Data Structures & **Up:** The Linux USB Subsystem **Previous:** Standard Device Requests
Contents **Index**

USB Transfers

This section will give an overview of all data structures, macros and functions related to data transfers on the bus. Further it will be explained how to actually set up, submit and process transfer requests.

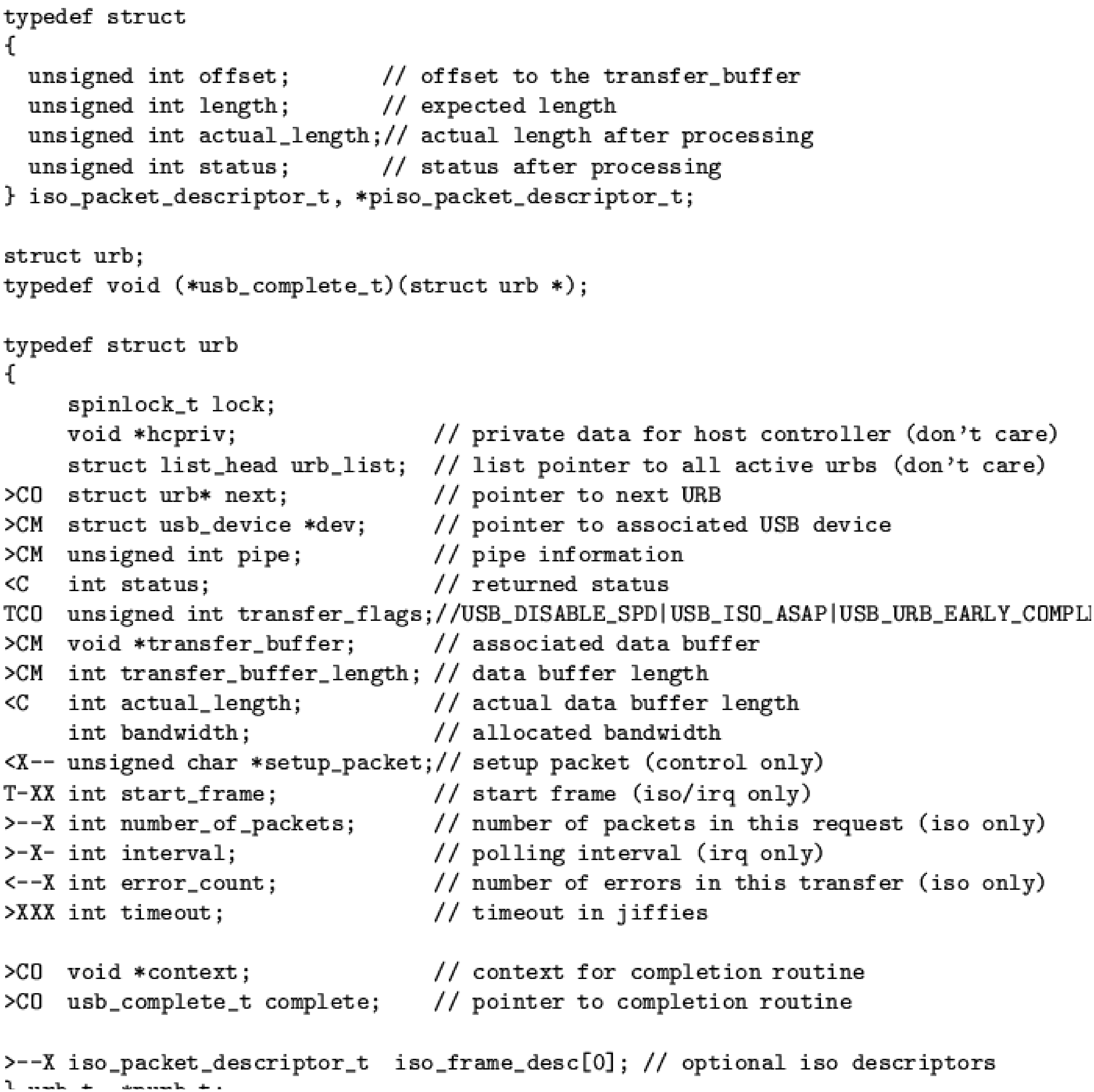
Subsections

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: URB Functions **Up:** USB Transfers **Previous:** USB Transfers **Contents** **Index**

Transfer Data Structures & Macros

The Linux USB subsystem uses only one data transfer structure called USB Request Block (URB). This structure contains all parameters to setup any USB transfer type. All transfer requests are sent asynchronously to the USB core and the completion of the request is signalled via a callback function.



```
typedef struct
{
  unsigned int offset;       // offset to the transfer_buffer
  unsigned int length;       // expected length
  unsigned int actual_length;// actual length after processing
  unsigned int status;       // status after processing
} iso_packet_descriptor_t, *piso_packet_descriptor_t;

struct urb;
typedef void (*usb_complete_t)(struct urb *);

typedef struct urb
{
     spinlock_t lock;
     void *hcpriv;                 // private data for host controller (don't care)
     struct list_head urb_list;    // list pointer to all active urbs (don't care)
>CO  struct urb* next;             // pointer to next URB
>CM  struct usb_device *dev;       // pointer to associated USB device
>CM  unsigned int pipe;            // pipe information
<C   int status;                   // returned status
TCO  unsigned int transfer_flags;//USB_DISABLE_SPD|USB_ISO_ASAP|USB_URB_EARLY_COMPL
>CM  void *transfer_buffer;        // associated data buffer
>CM  int transfer_buffer_length;   // data buffer length
<C   int actual_length;            // actual data buffer length
     int bandwidth;                // allocated bandwidth
<X-- unsigned char *setup_packet;// setup packet (control only)
T-XX int start_frame;              // start frame (iso/irq only)
>--X int number_of_packets;        // number of packets in this request (iso only)
>-X- int interval;                 // polling interval (irq only)
<--X int error_count;              // number of errors in this transfer (iso only)
>XXX int timeout;                  // timeout in jiffies

>CO  void *context;                // context for completion routine
>CO  usb_complete_t complete;      // pointer to completion routine

>--X iso_packet_descriptor_t  iso_frame_desc[0]; // optional iso descriptors
} urb_t, *purb_t;
```

```
>--X iso_packet_descriptor_t  iso_frame_desc[0]; // optional iso descriptors
} urb_t, *purb_t;
```

Figure 7:URB Structure

As shown in figure 7 the URB structure contains elements common to all transfer types (marked with C). Elements marked with > are input parameters, M means mandatory and O means optional. Elements marked with < are return values. Elements marked with T are transient parameters (input and output). All non common elements are marked on three columns which represent control, interrupt and isochronous transfers. A X marks this element to be used with the associated transfer type.

The URB structure might look confusing but this is just an overview of its versatility. There are several helping macros to setup the right parameters but first the common elements will be explained as they are very important.

- `dev` [mandatory input parameter]

  This element is a pointer to the usb_device structure (introduced in the framework function `probe` section 2.1.2).

- `pipe` [mandatory input parameter]

  The pipe element is used to encode the endpoint number and properties. There exist several macros to create an appropriate pipe value:

  - `pipe=usb_sndctrlpipe(dev,endpoint)`

    `pipe=usb_rcvctrlpipe(dev,endpoint)`

    Creates a pipe for downstream (snd) or upstream (rcv) control transfers to a given endpoint. The argument `dev` is a pointer to a usb_device structure. The argument `endpoint` is usually 0.

  - `pipe=usb_sndbulkpipe(dev,endpoint)`

    `pipe=usb_rcvbulkpipe(dev,endpoint)`

    Creates a pipe for downstream (snd) or upstream (rcv) bulk transfers to a given endpoint. The endpoint is of `1 <= endpoint <= 15` (depending on active endpoint descriptors)

  - `pipe=usb_sndintpipe(dev,endpoint)`

    `pipe=usb_rcvintpipe(dev,endpoint)`

    Creates a pipe for downstream (snd) or upstream (rcv) interrupt transfers to a given endpoint. The endpoint is of `1 <= endpoint <= 15` (depending on active endpoint descriptors)

  - `pipe=usb_sndisopipe(dev,endpoint)`

    `pipe=usb_rcvisopipe(dev,endpoint)`

Creates a pipe for downstream (snd) or upstream (rcv) isochronous transfers to a given endpoint. The endpoint is of 1 <= endpoint <= 15 (depending on active endpoint descriptors)

- `transfer_buffer` [mandatory input parameter]

  This element is a pointer to the associated transfer buffer which contains data transferred from or to a device. This buffer has to be allocated as a non-pageable contiguous physical memory block (simply use `void *kmalloc(size_t, GFP_KERNEL);`).

- `transfer_buffer_length` [mandatory input parameter]

  This element specifies the size of the transfer buffer in bytes. For interrupt and control transfers the value has to be less or equal the maximum packet size of the associated endpoint. The maximum packet size can be found as element `wMaxPacketSize` of an endpoint descriptor. Because there is no endpoint descriptor for the default endpoint 0 which is used for all control transfers the maximum packet size can be found as element `maxpacketsize` of the `usb_device` structure.

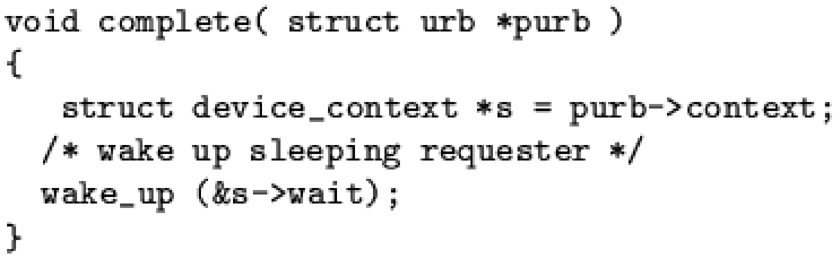
  Bulk transfers which are bigger than `wMaxPacketSize` are automatically split into smaller portions.

- `complete` [optional input parameter]

  As noted above the USB subsystem processes requests asynchronously. This element allows to specify a pointer to a caller supplied handler function which is called after the request is completed. The purpose of this handler is to finish the caller specific part of the request as fast as possible because it is called out of the host controller's hardware interrupt handler. This even implies all other restrictions that apply for code which is written for interrupt handlers.

- `context` [optional input parameter]

  Optionally a pointer to a request related context structure can be given. Figure 8 shows a simple completion handler.



```
void complete( struct urb *purb )
{
   struct device_context *s = purb->context;
  /* wake up sleeping requester */
  wake_up (&s->wait);
}
```

Figure 8:A simple completion handler

- `transfer_flags` [optional input parameter and return value]

  A number of transfer flags may be specified to change the behaviour when processing the transfer request.

  - `USB_DISABLE_SPD`

    This flag disables short packets. A short packet condition occures if an upstream request transfers less data than maximum packet size of the associated endpoint.

    - `USB_NO_FSBR`
    - `USB_ISO_ASAP`

      When scheduling isochronous requests this flag tells the host controller to start the transfer as soon as possible. If `USB_ISO_ASAP` is not specified a start frame has to be given. It is recommended to use this flag if isochronous transfers do not have to be synchronized with the current frame number. The current frame number is a 11 bit counter that increments every millisecond (which is the duration of 1 frame on the bus). Further documentation can be found in [4] sections 5.10.6 and 5.10.8.

    - `USB_ASYNC_UNLINK`

      When a URB has to be cancelled (see 2.3.2) it can be done synchronously or asynchronously. Use this flag to switch on asynchronous URB unlinking.

    - `USB_TIMEOUT_KILLED`

      This flag is only set by the host controller to mark the URB as killed by timeout. The URB status carries the actual error which caused the timeout.

    - `USB_QUEUE_BULK`

      This flag is used to allow queueing for bulk transfers. Normally only one bulk transfer can be queued for an endpoint of a particular device.

- `next` [optional input parameter]

  It is possible to link several URBs in a chain by using the next pointer. This allows you to send a sequence of USB transfer requests to the USB core. The chain has to be terminated by a NULL pointer or the last URB has to be linked with the first. This allows to automatically reschedule a number of URBs to transfer a continous data stream.

- `status` [return value]

  This element carries the status of an ongoing or already finished request. After successfully sending a request to the USB core the status is `-EINPROGRESS`. The successful completion of a request is indicated by 0. There exist a number of error conditions which are documented in section 3.1.

- `actual_length` [return value]

  After a request has completed this element counts the number of bytes transferred.

The remaining elements of the URB are specific to the transfer type.

- Bulk Transfers

  No additional parameters have to be specified.

- Control Transfers

- `setup_packet` [mandatory input parameter]

  Control transfers consist of 2 or 3 stages (see [4] sections 5.5, 8.5.2). The first stage is the downstream transfer of the setup packet. This element takes the pointer to a buffer containing the setup data. This buffer has to be allocated as a non-pageable contiguous physical memory block (simply use `void *kmalloc(size_t, GFP_KERNEL);`).

- Interrupt Transfers
  - `start_frame` [return value]

    This element is returned to indicate the first frame number the interrupt is scheduled.

  - `interval` [mandatory input parameter] This element specifies the interval in milliseconds for the interrupt transfer. Allowed values are `1 <= interval <= 255`. Specifying an interval of 0ms causes an one shot interrupt (no automatic rescheduling is done). You can find the interrupt interval as element `bInterval` of an endpoint descriptor for interrupt endpoints.

- Isochronous Transfers
  - `start_frame` [input parameter or return value]

    This element specifies the first frame number the isochronous transfer is scheduled. Setting the `start_frame` allows to synchronize transfers to or from a endpoint. If the `USB_ISO_ASAP` flag is specified this element is returned to indicate the first frame number the isochonous transfer is scheduled.

  - `number_of_packets` [mandatory input parameter]

    Isochronous transfer requests are sent to the USB core as a set of single requests. A single requests transfers a data packet up to the maximum packet size of the specified endpoint (pipe). This element sets the number of packets for the transfer.

  - `error_count` [return value]

    After the request is completed (URB status is != `-EINPROGRESS`) this element counts the number of errorneous packets. Detailed information about the single transfer requests can be found in the `iso_frame_desc` structure.

  - `timeout` [input parameter] A timeout in jiffies can be specified to automatically remove a URB from the host controller schedule. If a timeout happens the transfer flag `USB_TIMEOUT_KILLED` is set. The actual transfer status carries the USB status which caused the timeout.

  - `iso_frame_desc` [mandatory input parameter]

    This additional array of structures at the end of every isochronous URB sets up the transfer parameters for every single request packet.

    - `offset` [mandatory input parameter]

      Specifies the offsetaddress to the `transfer_buffer` for a single request.

- `length` [mandatory input parameter]

  Specifies the length of the data buffer for a single packet. If `length` is set to 0 for a single request the USB frame is skipped and no transfer will be initiated. This option can be used to synchronize isochronous data streams (specified in [4] section 5.6).

- `actual_length` [return value]

  Returns the actual number of bytes transferred by this request.

- `status` [return value]

  Returns the status of this request. Further documentation can be found in section 3.1.

Next Up Previous Contents Index
Next: URB Functions **Up:** USB Transfers **Previous:** USB Transfers **Contents** **Index**

Detlef Fliegl
2001-01-08

URB Functions

There are four functions of the USB core that handle URBs.

- `purb_t usb_alloc_urb(int iso_packets);`

  Whenever a URB structure is needed this function has to be called. The argument `iso_packets` is used to specify the number of `iso_frame_desc` structures at the end of the URB structure when setting up isochronous transfers. If successful the return value is a pointer to a URB structure preset to zero otherwise a NULL pointer is returned.

- `void usb_free_urb (purb_t purb);`

  To free memory allocated by `usb_alloc_urb` simply call this function.

- `int usb_submit_urb(purb_t purb);`

  This function sends a transfer request asynchronously to the USB core. The argument `purb` is a pointer to a previously allocated and initialized URB structure. If successful the return value is 0 otherwise an appropriate error code is returned (see section 3.1). *The function returns always non-blocking and it is possible to schedule several URBs for different endpoints without waiting. On isochronous endpoints it is even possible to schedule more URBs for one endpoint.* This limitation is caused due to error handling and retry mechanisms of the USB protocol (see [4] section 8.5)

- `int usb_unlink_urb(purb_t purb);`

  This function cancels a scheduled request before it is completed. The argument `purb` is a pointer to a previously submitted URB structure. The function can be called synchronously or asynchronously depending on the transfer_flag `USB_ASYNC_UNLINK` (see 2.3.1). Synchronously called the function waits for 1ms and must not be called from an interrupt or completion handler. The return value is 0 if the function succeeds. Asynchronously called the function returns immediately. The return value is -EINPROGRESS if the function was successfully started. When calling `usb_unlink_urb` the completion handler is called after the function completed. The URB status is marked with -ENOENT (synchronously called) or -ECONNRESET (asynchronously called).

`usb_unlink_urb` is also used to stop an interrupt transfer URB. As documented in sections 1.2.2, 2.3.1 interrupt transfers are automatically rescheduled. Call `usb_unlink_urb` even for ``one shot interrupts''.

Next Up Previous Contents Index

Next: URB Macros **Up:** USB Transfers **Previous:** Transfer Data Structures & **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Compatibility Wrappers **Up:** USB Transfers **Previous:** URB Functions **Contents** **Index**

URB Macros

To initialize URB structures for different transfer types there exist some macros:

- `FILL_CONTROL_URB(purb, dev, pipe, setup_packet, transfer_buffer, transfer_buffer_length, complete, context);`
- `FILL_BULK_URB(purb, dev, pipe, transfer_buffer, transfer_buffer_length, complete, context);`
- `FILL_INT_URB(purb, dev, pipe, transfer_buffer, transfer_buffer_length, complete, context, interval);`
- `FILL_CONTROL_URB_TO();`
- `FILL_BULK_URB_TO();`

The macros are self explaining - more documentation can be found in the include file usb.h.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Examples **Up:** USB Transfers **Previous:** URB Macros **Contents** **Index**

Compatibility Wrappers

The USB core contains a number of higher level functions which were introduced as compatibility wrappers for the older APIs. Some of these functions can still be used to issue blocking control or bulk transfers.

- `int usb_control_msg(struct usb_device *dev, unsigned int pipe, __u8 request, __u8 requesttype, __u16 value, __u16 index, void *data, __u16 size, int timeout);`

  Issues a blocking standard control request. The arguments are according to [4] section 9.3. A timeout in jiffies has to be specified. If successful the return value is a positive number which represents the bytes transferred otherwise an error code is returned.

- `int usb_bulk_msg(struct usb_device *usb_dev, unsigned int pipe, void *data, int len, unsigned long *actual_length, int timeout);`

  Issues a blocking bulk transfer. The standard arguments should be self explaining. `actual_length` is an optional pointer to a variable which carries the actual number of bytes transferred by this request. A `timeout` in jiffies has to be specified.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Reference **Up:** The Linux USB Subsystem **Previous:** Compatibility Wrappers **Contents** **Index**

Examples

A sample device driver is the dabusb driver which is part of the latest kernel tree. The driver covers these topics:

- Supporting multiple devices
- Claiming an interface
- Setting configuration and alternate settings
- Submitting control and bulk URBs
- Reading an isochronous data stream
- Allowing hot unplug

You can find further information and updates on [3], [2]

Now some code fragments will follow to show how to actually program different transfers.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Error Codes **Up:** Programming Guide for Linux **Previous:** Examples **Contents** **Index**

Reference

Subsections

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Error codes returned by **Up:** Reference **Previous:** Reference **Contents** **Index**

Error Codes

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

Subsections

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: URB Error Codes **Up:** Error Codes **Previous:** Error Codes **Contents** **Index**

Error codes returned by usb_submit_urb

- Non USB specific

0	URB submission successful
-ENOMEM	No memory for allocation of internal structures

- USB specific

`-ENODEV`	Specified USB-device or bus doesn't exist
`-ENXIO`	URB already queued
`-EINVAL`	a) Invalid transfer type specified (or not supported)
	b) Invalid interrupt interval (0<=n<256)
	c) More than one interrupt packet requested
`-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`	Too much ISO frames requested (currently uhci >900)
`-EPIPE`	Specified pipe-handle is already stalled
`-EMSGSIZE`	Endpoint message size is zero, do interface/alternate setting

Next Up Previous Contents Index
Next: URB Error Codes **Up:** Error Codes **Previous:** Error Codes **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index
Next: Error Codes returned by **Up:** Error Codes **Previous:** Error codes returned by **Contents** **Index**

URB Error Codes

- These error codes are returned in `urb->status` or `iso_frame_desc[n].status`:

0	Transfer completed successfully
`-ENOENT`	URB was canceled by unlink_urb
`-EINPROGRESS`	URB still pending, no results yet (actually no error until now)
`-EPROTO`	a) Bitstuff error
	b) Unknown USB error
`-EILSEQ`	CRC mismatch
`-EPIPE`	a) Babble detect
	b) Endpoint stalled
`-ENOST`	Buffer error
`-ETIMEDOUT`	Transfer timed out, NAK
`-ENODEV`	Device was removed
`-EREMOTEIO`	Short packet detected
`-EXDEV`	ISO transfer only partially completed look at individual frame status for details
`-EINVAL`	ISO madness, if this happens: Log off and go home

Next Up Previous Contents Index
Next: Error Codes returned by **Up:** Error Codes **Previous:** Error codes returned by **Contents** **Index**

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Bibliography **Up:** Error Codes **Previous:** URB Error Codes **Contents** **Index**

Error Codes returned by USB Core Functions

- `usb_register()`:

  `-EINVAL` Error during registering new driver.

- `usb_terminate_bulk()`:

  `-ENODEV` URB already removed.

- `usb_get_*/usb_set_*()`:

  All USB errors (submit/status) can occur.

Detlef Fliegl
2001-01-08

Next Up Previous Contents Index

Next: Index **Up:** Programming Guide for Linux **Previous:** Error Codes returned by **Contents** **Index**

Bibliography

1

http://www.usb.org, Universal Serial Bus Implementers Forum

2

http://www.linux-usb.org, Linux USB Developer and Support information.

3

http://usb.cs.tum.edu, Linux USB Developer Pages

4

Universal Serial Bus Specification Compaq, Intel, Microsoft, NEC, Revision 1.1, September 23, 1998

5

Universal Serial Bus Common Class Specification Systemsoft Corporation, Intel Corporation, Revision 1.0 December 16, 1997

6

Device Class Definition for Human Interface Devices (HID) Firmware Specification, Version 1.1, Universal Serial Bus (USB),July 4, 1999

7

Intel Universal Host Controller Interface (UHCI) Design Guide, Revision 1.1, March 1996

8

Linux Device Drivers, 1st Edition, Alessandro Rubini, February 1998

9

http://selva.dit.upm.es/jmseyas/linux/kernel/hackers-docs.html, Index of Documentation for People Interested in Writing and/or Understanding the Linux Kernel, Juan-Mariano de Goyeneche

Detlef Fliegl
2001-01-08

About this document ...

Programming Guide for Linux USB Device Drivers

This document was generated using the **LaTeX**2HTML translator Version 99.1 release (March 30, 1999)

The command line arguments were:
latex2html `-local_icons -rootdir usbdoc -up_url http://usb.cs.tum.edu -up_title USB Developer Pages usbdoc.tex`

The translation was initiated by Detlef Fliegl on 2001-01-08

Detlef Fliegl
2001-01-08

Institut für Informatik TU-München

Lehr- und Forschungseinheit Informatik X

Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechner

Prof. Dr. A. Bode

Universal Serial Bus Development for Linux

G. Acher & D. Fliegl & T. Sailer & R. Weissgärber

People

- Dipl. Inform. Georg Acher (UHCI driver, URB specification, USB-Core)
- Dipl. Inform. Detlef Fliegl (UHCI driver, URB specification, DABUSB driver)
- Dipl. Ing. Thomas Sailer (UHCI driver, URB specification, Audio driver)
- Dipl.-Ing. Roman Weissgärber (OHCI driver, UHCI virtual root hub)

USB-Team, $Date: 2002/03/23 19:04:22 $

powered by SuSE Linux

Institut für Informatik TU-München

Lehr- und Forschungseinheit Informatik X

Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechner

Prof. Dr. A. Bode

Universal Serial Bus Development for Linux

G. Acher & D. Fliegl & T. Sailer & R. Weissgärber

Links

- USB Impl. Forum
- USB-IF Devel. page
- USB OpenHCI Spec.
- USB Vendor List

- Linux USB
- Linux HOWTO: USB (dynamine.net)
- Linux HOWTO: USB (cheek.com)
- Linux-USB device overview
- Laptops with USB Ports working with Linux
- Linux-USB backport patch
- Linux USB Email Archives (electricrain.com)
- linux-usb email archives (suse.com)
- Linux USB (Rewrite) Project
- Programming Guide for Linux USB Device Drivers
- Index of /usb (dynamine.net)
- OmniVision -- Universal Serial Bus (USB) Application Products
- Linux USB for DC-2xx Cameras
- Linux-USB modules for Philips webcams
- Linux USB Scanner Driver (DNelson)
- Linux-USB SW (G.Smith, Suite9)
- uusbd/ (Cal. mirror)
- uusbd: Iñaky/

- uusbd: Iñaky/
- Using the ActiveWire USB board with Linux

- Cypress Semiconductor
- Anchor Chips (Cypress)
- USB EZ-Link Instant Network (Cypress)
- ActiveWire: USB
- ADMtek: USB NIC
- Lucent: USB Product Doc.
- NetChip (USB)
- OPTi: USB Solutions
- TI: USB

- USB: allUSB news & info.
- USB Cable Products
- USB Catalog
- USB Central: Info. (IOTech)
- USB Gamepad App. Note
- USB Gear OnLine Catalog
- USB Links
- USBmax OnLine Catalog
- USBnews
- USB Pinouts (Starmount)
- USB products (cablesnmor.com)
- USBStuff OnLine Catalog
- USB Workshop
- USB Design By Example (book)

- USB PnP IDs (MS)
- USB Tech. (MS)
- HW Dev. Stds./Specs. (BIOS, 1394, PCCard, PCI, PnP, USB) (MS)
- Free BSD USB home page
- Apple USB Developers

USB-Team, $Date: 2002/03/23 19:04:22 $

powered by SuSE Linux

Institut für Informatik TU-München

Lehr- und Forschungseinheit Informatik X

Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechner

Prof. Dr. A. Bode

http://usb.cs.tum.edu

Universal Serial Bus Development for Linux

G. Acher & D. Fliegl & T. Sailer & R. Weissgärber

Download

You can find all latest sources in the current 2.4/2.5 kernel tree.

Our download archive contains:

- **usbd** USB user space daemon (start of development)
- **usbdoc** USB Programming Guide
- **usbstress** USB stress test package
- **usbutils** USB utilities to verbose the /proc/bus/usb entries

Latest News

- (2000-06-01) You can find all latest sources in the current 2.4.X kernel tree.
- (2000-03-14) UHCI: fixed bandwidth reclamation for intel chipsets, added timeouts, added bulk queueing. Added driver for Prolific USB Net driver 'plusb'.
- (2000-02-18) UHCI: now containing bandwidth reclamation (as an option). The reclamation is intelligent, as it disables the loop if the urb (bulk|control) is pending longer than 50ms, thus avoiding PCI congestion. Switched to "breadth first"-descriptor processing (bulk|control) for a fair distribution with the reclamation loop (can be adjusted via defines).
- (2000-02-02) UHCI: remove of pending URBs at disconnect, hardware race fix, simplified locking code, variable namespace cleanup USBDEVFS: fixed async bulk cleanup code DABUSB: plug/reload init sequence fix.
- (2000-01-17) USB package merged with latest patches of 2.3.40-pre4. New major patch against 2.3.40-pre4 available. See detailed changes here.
- (2000-01-08) USB package merged with latest patches of 2.3.38, 2.2.X backport fixes applied.
- (2000-01-06) USB device filesystem added, Cleanups in audio, usbcore, uhci, dabusb etc., Minor fixes in ohci, uhci
- (1999-12-31) Programming Guide for Linux USB Device Drivers added
- (1999-12-27) uhci.c interrupt transfer fix, usb.h patch applied, minor fixes in audio, mouse
- (1999-12-21) New CVS web frontend. Added T. Sailer's usbutils and usbd.
- (1999-12-20) Fixed SMP issues and a problem with short packet handling in uhci, blocking mouse read, fixed another oops in audio during hot unplug, ohci isochronous tranfers work with cpia and audio, added multiple host controllers patch for proc_usb
- (1999-12-17) Minor fixes in proc_usb.c, mouse.c, uhci.c, beautified code in dabusb.c
- (1999-12-16) Fixed many memory leaks and OOPS in usbcore and hub driver, UHCI fair queueing between LS/FS-devices, APC UPS fix, fixed cleanup after failing in initialization, there definitly is NO memory leak in UHCI, OHCI correct error msg length != cnt in td_submit_urb for int TDs, audio:Fix crash when reading from write-only device and writing to read-only device, fixed hot-unplug in mixer, (still hot unplug problems reported), mouse: hot unplug fix
- (1999-12-15) Fixed bulk compatibility wrapper in usbcore, audio driver hot unplug fix, removed old root hub code from UHCI driver and fixed SMP race when unlink_urb is called
- (1999-12-14) Fixed Bugs in usbcore, UHCI is SMP safe (24h test passed), fixed control transfer problem
- (1999-12-13a) UHCI-driver respects SMP locks (further testing necessary)
- (1999-12-13) fixed hub code to work with philips webcam (increased setup time)
- (1999-12-12) new error codes, synch to 2.3.31, included hub, error code documentation
- (1999-12-9) Clean audio unplugging, hot unload of uhci, ohci hub cleanup

- (1999-12-8) Hub driver fix (audio driver works again)
- (1999-12-6) EP-parsing, small fixes
- (1999-12-4) Cleaned up error codes, SP-fix for CTRL-IN (1st try)
- (1999-12-3) Included virtual root hub (by Roman Weissgaerber), code beautified with indent
- (1999-11-27) Attempt to reduce dropouts for iso under heavy load, SMP-clean lock for unlink_urb
- (1999-11-22a) Fixed auto-resubmitting of URBs
- (1999-11-22) Small backport fixes (now supporting: i386 and axp Linux 2.2.X and 2.3.X kernels)
- (1999-11-21) Backport to 2.2.X kernels, small fixes
- (1999-11-16) Fixed string descriptors, one-shot-interrupts.
- (1999-11-14) Adaption of usbcore to 2.3.27, small fixes, usb_scsi changed to new uhci, but not tested.
- (1999-11-10) The compatibility wrapper function of the core for irq is now fixed.
- (1999-11-10) Hub driver works.

USB-Team, $Date: 2000/03/14 22:29:00 $

SuSE Logo
The Linux Experts

Deutschland,
Österreich
und Schweiz

Kontakt | Sitemap | Links | Produkt-Registrierung

Suche [ ] it

Country/Language it

- Security Announcements
- Maintenance Web
- Supportdatenbank
- Hardwaredatenbank
- Produkt-Registrierung
- Zertifizierte Hardware

SuSE is a member of

UnitedLinux

Initiative D21

Privatkunden

- SuSE Linux Produkte
- SuSE PRESS Bücher
- Fanartikel
- Info & Support
- Downloads
- Online Shop
- Testberichte

Geschäftskunden

- Produkte & Lösungen
- Professional Services
- Maintenance & Support
- Training
- Zertifizierungen
- Testberichte

Partner

- Partner werden
- Partner suchen
- Ich bin Partner
- Partnernews

Über SuSE

- Das Unternehmen
- Kontakt
- Presse
- Veranstaltungen
- Referenzkunden
- SuSE Education Programm
- Jobs & Karriere

Neu: SuSE Linux Desktop für Unternehmen

News

16.07.2003: SuSE Linux Enterprise Server erreicht Spitzenwert bei Oracle Benchmark

14.07.2003: Petra Heinrich von SuSE in den Vorstand des Linux-Verbands gewählt

14.07.2003: Bundesinnenministerium stellt Migrationsleitfaden für Open-Source-Software vor

14.07.2003: Über 19.500 Besucher auf dem LinuxTag

09.07.2003: Kostenloser Download: SuSE Linux 8.2 Beta für AMD Athlon 64

Linux is a registered trademark of Linus Torvalds

This server is powered by NPS.

Page created: 12.06.2003 12:17 MET DST webmaster@suse.de ▶ Impressum & Datenschutz

LRR-TUM-Logo

Fakultät für Informatik

der Technischen Universität München

Informatik X: Rechnertechnik und Rechnerorganisation / Parallelrechnerarchitektur

Prof. Dr. Arndt Bode , Prof. Dr. Hans Michael Gerndt

Home | Adressen | Personen | Forschung | Lehrveranstaltungen

Suche

Research Groups and Projects

Prof. Dr. M. Gerndt
Professor

Prof. Dr. A. Bode
Professor

Dr. W. Karl
Senior Scientist

PARALLEL AND DISTRIBUTED ARCHITECTURES AND APPLICATIONS

Programming Environments and Tools

Dr. R. Wismüller

Applications

Dr. P. Luksch

Architectures

Dr. C. Trinitis

Hardware Laboratory and Computer Assisted Training

B. Piochacz

Secretary

Computing Environment

Technical Staff

Attention: Our webserver is being updated, please check our **new LRR-Webserver** for latest news!

General Information

- Addresses
- Staff
- Publications
- Events
- How to get here
- Stellenangebote

Info für Studenten

- Vorlesungen (Informatik)
- Lectures (CSE Program)
- Seminare
- Praktika
- Diplomarbeiten
- Systementwicklungsprojekte (SEP)
- **NEU!!!** Auslandskontakt: ENS Lyon
- Hiwi Jobs

Research Groups

- Programming environments and tools
- Applications
- Architectures
- Projects

Miscellaneous

- GI/ITG Fachgruppe APS+PC
- LRR-TUM houses KONWIHR's Munich office
- BIOINFORUM - Bioinformatik 2001

Internal

- Public information
- Non-public information

HTDIG Search

Webmaster
$Id: index.html,v 1.87 2003/07/17 10:10:27 fliegl Exp $

Fakultät für Informatik

der Technischen Universität München

HOME | STUDIUM | FORSCHUNG | FAKULTÄT | DIENSTE | INFORMATION | ALUMNI

Login **Neuer Benutzer** Suche - Hilfe

STUDIUM
Schulportal
Studienberatung
Internationales
Informatik
Bioinformatik
Wirtschaftsinformatik
Vorlesungen
Skripte

FORSCHUNG
Schwerpunkte
Projekte
Publikationen
Stellenangebote

FAKULTÄT
Dekanat
Lehrstühle
Professoren
MitarbeiterInnen
Festveranstaltungen
Kolloquium

DIENSTE
Rechenbetrieb
Bibliotheken
Servicebüros

INFORMATION
Adresse, Anfahrt
LEO, Links
Jobs

ALUMNI
Adressbuch
Jahrgangslisten

Impressum

Willkommen an der Fakultät für Informatik!

STUDIUM

Die Vorlesungszeit des Sommersemesters 2003 beginnt am Montag, den 7.4.2003 und endet am Freitag, den 11.7.2003. Auf dieser Website finden Sie sowohl eine Übersicht aller Lehrveranstaltungen als auch Informationen zur Bewerbung um einen Studienplatz.

Ab dem Wintersemester 2002/2003 bietet die Fakultät auch ein spezielles Förderprogramm für Begabte. Studierende im ersten und dritten Fachsemester können sich bis zum 1. Oktober, 2002 bewerben.

FAKULTÄT

Die Fakultät ist nach dem Ende des Sommersemesters 2002 von der Innenstadt auf den Campus Garching umgezogen. Bitte beachten Sie die neue Adresse und die neuen Telefonnummern.

DREHSCHEIBE

Der WAP-Zugang zu einem Teil der Informationen und Funktionalitäten der

Current Events

- ARENA Software Engineering Praktikum: Abschlusspräsentation [18.07.2003 10:15]
- Vortrag 'Degrees of Complexity in Polynomial Ideals' [18.07.2003 13:00]
- Sonderkolloquium: Ein didaktisches System fuer objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II [21.07.2003 10:00]

Neue Mitteilungen

- Stellenausschreibung: Wissenschaftliche Hilfskraft
- Werkstudentenjob
- Werkstudentenjob
- Programm in der Neurobiologie
- Einteilung Praktika/Programmierpraktika
- Vortrag 'Degrees of Complexity in Polynomial Ideals' [18.07.2003 13:00]
- Verlegung Sprechstunde Studienberatung
- SEP: Aufbau und Evaluierung eines MMS Gateway für 2G und UMTS als Webservice
- ARENA Software Engineering Praktikum: Abschlusspräsentation [18.07.2003 10:15]

... mehr Ankündigungen

Feedback

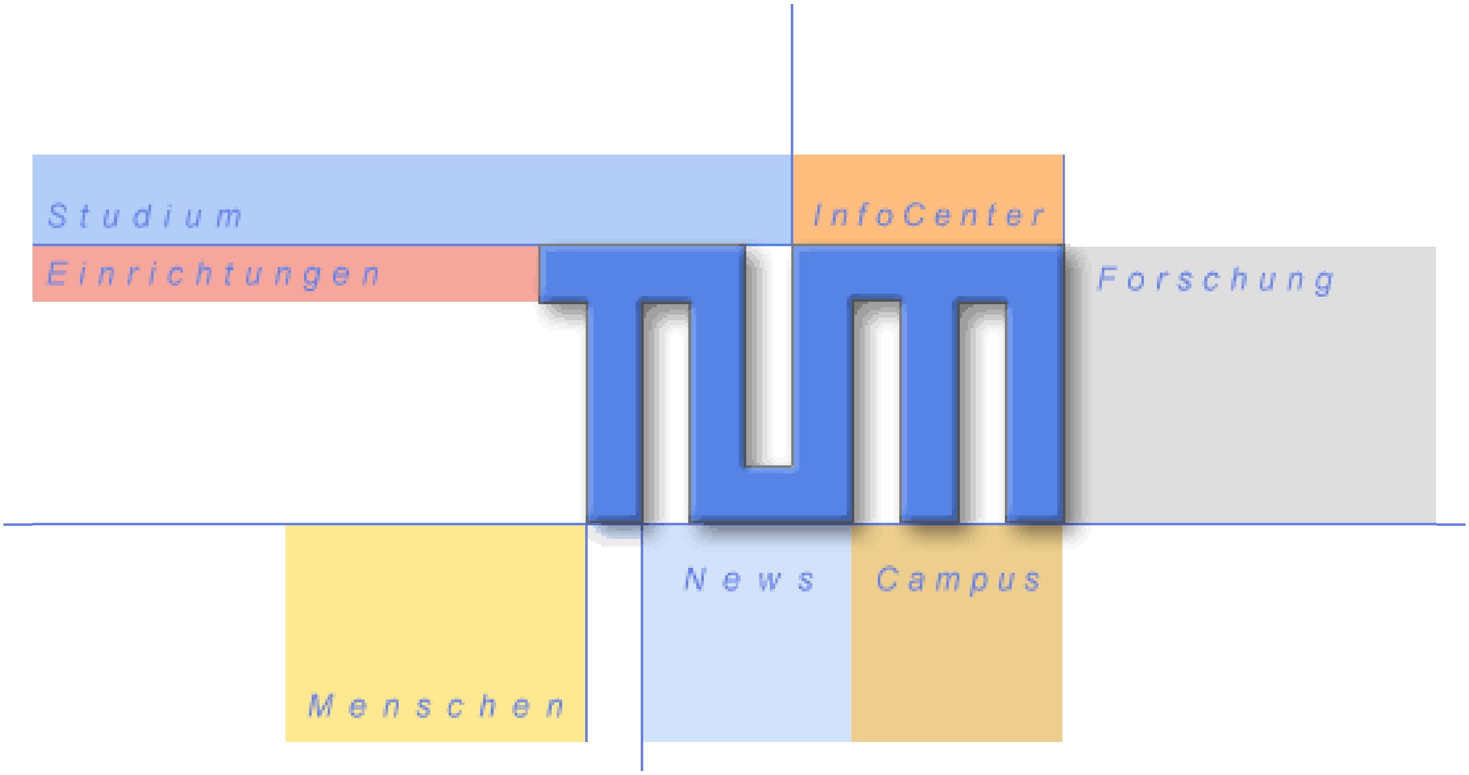
Drehscheibe ist jetzt im Beta-Test. Mehr dazu unter /doku/wap!

Heute in den Mensen:

- Mensa des Studentenwerks
- Cafeteria im FMI-Bau

Suche nach [ ] in [Ankündigungen] [Suchen]

Student in München? Dann schaut doch mal auf www.studiosity.de vorbei ...



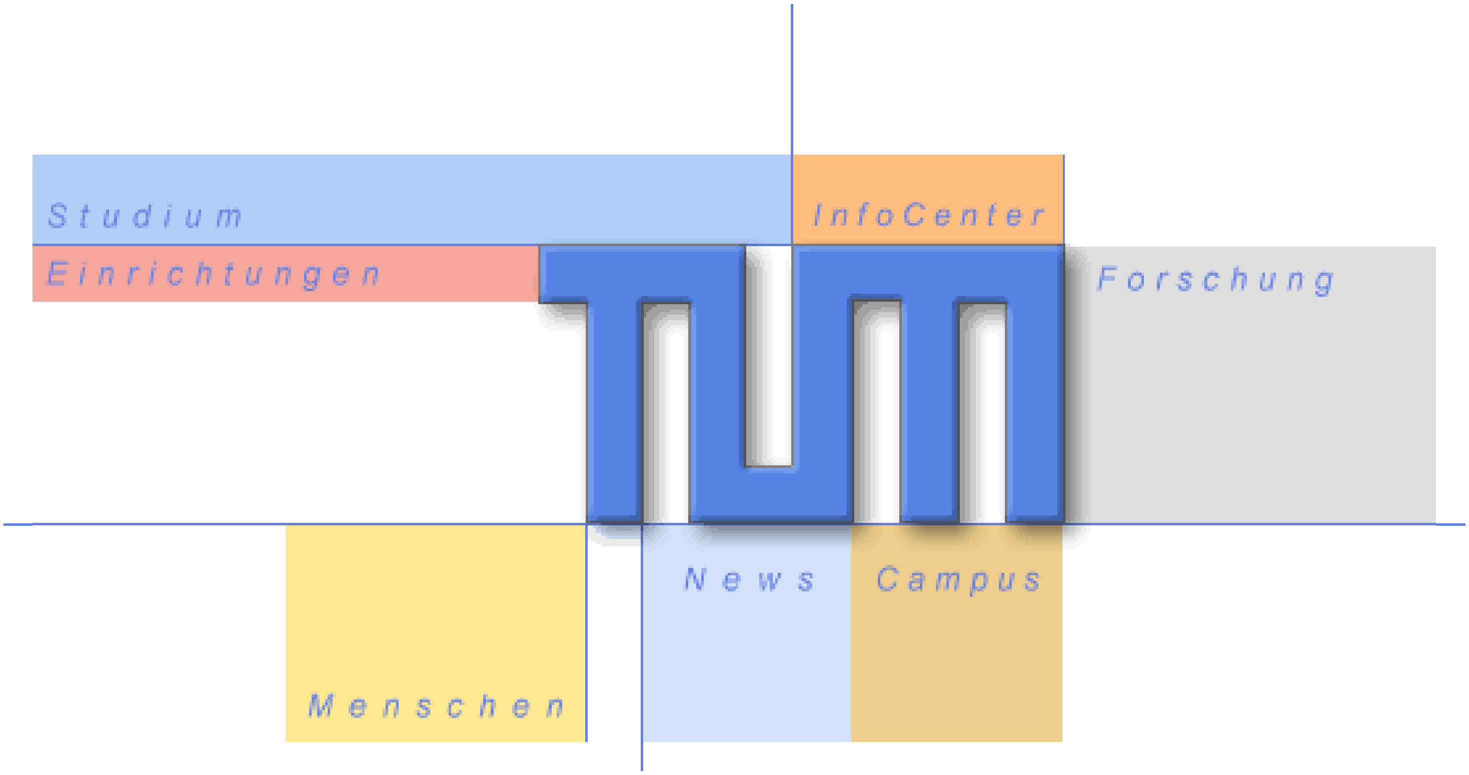
○ english version

TECHNISCHE UNIVERSITÄT MÜNCHEN

Hausanschrift: Arcisstr. 21, 80333 München · Briefanschrift: 80290 München, Tel. (089) 289-01 · Fax (089) 289-22000

QuickClicks:

Fakultäten	UnivIS	SiteMap
Presse	Studiengänge	Suche
International	Veranstaltungen	CampusFinder
Alumni	SAP@TUM	Kontakt



○ english version

quick clicks ○

TECHNISCHE UNIVERSITÄT MÜNCHEN

Hausanschrift: Arcisstr. 21, 80333 München · Briefanschrift: 80290 München, Tel. (089) 289-01 · Fax (089) 289-22000

QuickClicks:

- Fakultäten
- Presse
- International
- Alumni
- UnivIS
- Studiengänge
- Veranstaltungen
- SAP@TUM
- SiteMap
- Suche
- CampusFinder
- Kontakt

Prof. Dr. Arndt Bode

Zimmer:	**01.04.054**
Telefon:	**+49 - 89 - 289 - 17654**
Fax:	**+49 - 89 - 289 - 17662**
Email:	***bode@in.tum.de***

Anschrift: **Technische Universität München**
Institut für Informatik, I10
Lehrstuhl für Rechnertechnik und Rechnerorganisation
Boltzmannstr. 3
85748 Garching b. München

Research Groups and Projects

- **Publications** (pdf / html)
- **Talks** (pdf / html)
- **Editorials** (pdf / html)
- **Program Committees** (pdf / html)

Prof. Dr. Arndt Bode, 2003-04-24

Universal Serial Bus

Search | Home | Channel | Press | Developers | Members | Products

Shopping for USB
USB Features
USB FAQ
About USB-IF, Inc.

USB IN THE NEWS

USB Nomenclature: Recent discussion about the various data transfer rates in the USB 2.0 standard.

Hi-Speed USB Plays 'Show and Tell' at Intel Developer Forum With First Camcorder, Native OS Support and Other Milestones (pdf, 17k)

No. 1 Peripheral Interface Takes Center Stage at Dev Con
(pdf, 17k)

Intel Releases USB 2.0 Enhanced Host Controller Interface 1.0 Specification, EHCI Compliance-Testing Program
(pdf, 78k)

FEATURED PRODUCT

Company: Belkin Components

Product: Belkin Hi-Speed USB 2.0 Video Bus

Details: Capture video from your camcorder, PC camera, and VCRs.

Get your products featured here - find out how!

UPCOMING USB EVENTS

INSTANT, NO HASSLE CONNECTIONS

Universal Serial Bus (USB) connects more than computers and peripherals. It has the power to connect you with a whole new world of PC experiences.

USB is your instant connection to the fun of digital photography or the limitless creative possibilities of digital imaging. You can use USB to connect with other people through the power of PC-telephony and video conferencing. Once you've tried USB, we think you'll grow quite attached to it!

Having trouble downloading or printing the pdfs on this page? Download the latest version of the free Acrobat Reader.

Site sponsored by USB Implementers Forum, Inc., creators of USB technology. About Us | Privacy Statement

KAVI® *where .orgs work*

UNIVERSAL SERIAL BUS

Search | Home | Channel | Press | Developers | Members | Products

- Hi-Speed USB
- USB On-The-Go
- Tools
- USB-IF eStore
- Documents
- USB-IF Compliance Program
- USB FAQ
- Events
- Discussion Forum
- Join USB-IF, Inc.
- Retail Discussion Groups

Home > **Developers**

USB-IF Developers Area

Start using the USB Logo Now!

Download the USB-IF Trademark License Agreement and Usage Guidelines for the USB-IF Logo. The license agreement must be signed to access Logo artwork and obtain the right to use the Logo with products that pass USB-IF compliance testing.

The agreement necessary for gaining access to the graphics approved for linking to the usb.org web site are also available.

Hi-Speed USB

Find out about Hi-Speed USB. Included are Hi-Speed USB backgrounders, a list of most common questions and answers, the USB 2.0 Adopters Agreement, current press releases, and more.

On-The-Go

The On-The-Go Supplement addresses the need for mobile interconnectivity between portable devices when a PC is not available. Learn more about the On-The-Go Supplement to USB 2.0 Specification and other On-The-Go information in this section.

Tools

Download the most current installation utilities.

WHAT'S NEW

USB High Speed Electrical Tool

Provided by Intel®, the USBHSET.exe self-extracting file contains instructions and an installation utility that outlines testing procedures for electrical signal quality in high-speed host, hub and device testing. Intel software is also included.

USB Packaging and Naming Recommendations

Find out about Packaging Recommendations

USB-IF Certified High Speed Test Facilities

Check here for the most current list of labs certified to do high speed testing.

Events

Find out about upcoming USB events:

USB-IF eStore

USB Functional Compliance Test Devices and USB 2.0 Specifications are now available for purchase from the USB-IF. USB-IF member companies are eligible to receive discounted rates. Orders will be fulfilled in the order they were received.

Documents

Download the most current revision of the USB spec, conference presentations (1998 to 2001), whitepapers, and compliance workshop checklists.

Compliance Program

The USB specification defines the product design targets at the level of interfaces and mechanisms. The USB-IF has instituted a Compliance Program that provides reasonable measures of acceptability.

How to Join

Do you want to join the USB-IF? You'll find the Implementers Forum members search here, along with the USB-IF Logo and usage guidelines, and the members information maintenance page (including instructions).

Getting a Vendor ID

If you are a new USB product developer looking to get a vendor ID for your company, there are two preferred options for doing this.

Discussion Groups

USB developers: Use this technical forum in HTML format to discuss USB issues.

Tech FAQ

This FAQ answers most of the commonly asked questions about

Everything from trade show participation to compatibility workshop schedules to registration forms!

Also, be sure to check out photographs of past USB-IF events in our Photo Gallery.

USB and USB-related products and developments.

OpenHCI -- Open Host Controller Interface Specification for USB

Preliminary Publication Date:	September 1995
Version 1.0 Publication Date:	December 1995
Revision 1.0a Date:	October 1996

New OpenHCI and USB information coming soon to this page!

This document is the latest preliminary revision of the OpenHCI (Open Host Controller Interface) Specification, Rev. 1.0a released to the public in October 1996. This version replaces Rev. 1.0 released in December 1995.

This revision of the specification, Rev. 1.0a, is provided "as is" with no warranties whatsoever including any warranty of merchantability, fitness for any particular purpose, or any other warranty otherwise arising out of any proposal, specification, or example.

Download a self-extracting version of the Word document (256KB).

If you need the Microsoft Word for Windows Viewer, you can get it here.

Download this document in Adobe Acrobat pdf format (653 KB).

If you need the Adobe Acrobat Reader, you can get it here.

privacy statement
legal notices

Mirrors:
[de|it|us]

Linux USB

Sections

Introduction
FAQs
Device Support
Tools
Bitkeeper
Mailing Lists
Links
Beanie Award
Thanks
Contacts

Welcome to the home of the Linux USB Project

This web site was created to serve as a central point of information for USB support under Linux. We've released our first press release.

Information on this web site

The information on this web site mostly revolves around the kernel USB stack originally coded by Linus:

- Kernel USB stack (aka Linus' Alternative USB stack)
  This stack was originally developed by Linus Torvalds as an alternative USB stack for Linux. Since his original announcement, many other people have submitted patches to fix various bugs and add support for other features and was first introduced into the main kernel tree with kernel v2.2.7.

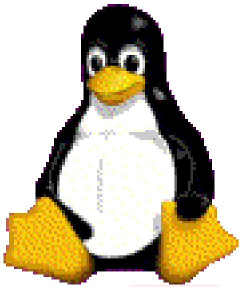
  NOTE: USB development is now being done in the 2.5 kernel tree. A lot was done in the 2.4 kernel and some features from 2.5 will be backported there. Much of USB support has been backported and is available starting with 2.2.18. It is listed a few lines below. USB-Storage is not supported in the 2.2 line of kernels.

Commercial Driver Support Requests

Results of email to open source the Philips Webcam driver.

Other helpful links

Linux USB Guide[html|ps]
Working devices list ---> **[Check here or on the linux-usb mailing list for device support.]**
Kernel 2.5 Todo list
Linux and USB 2.0



SourceForge Logo

USB Vendor/Device IDs list

USB Device Number mappings (major:minor)

USB Programming Guide

usbstress package

More USB links (standards, drivers, products, Other USB stacks, etc.)

User/help list is now linux-usb-users mailing list

Linux-USB SourceForge page

A few presentations are available

The process is now open to apply for funds from the Beanie Award

LiNUX-USB device overview

[Devices](#) [Drivers](#) [Controllers](#) [Links](#) [Add/Edit](#) [Search](#) Quick Search! [Register](#) [Contact](#) [Home](#)

Not logged in

Login to get access to the new bookmark feature!

Username:

Password:

Login!

You may register here, if you have no login yet.

News

Manage devices

2003-06-09

Management of device information is now only possible via the new form available after you logged into your account. Before you can manage your devices, you must migrate them to your account. This is done by logging in and then clicking on the link "migrate devices" - you can then enter the credentials you used to type on the old form (your email address plus a password). If you have any problems with the new system please do not hesitate to ask me.

Change profile information

2003-05-11

Finally it is now possible to change your account profile information. Just click on "Manage my profile" if you want to change your email address or name. You may even delete your account.

Hardware Upgrade

2002-11-07

In the next few days, there will be a hardware upgrade for this server (http://www.qbik.ch). You may therefore notice a short downtown for this service. The old hardware (a Compaq 486 PC with Pentium Overdrive (83 MHz!!!) and 32 MB RAM) will be replaced by a HP PC with PIII 450 MHz and 256 MB RAM running Mandrake Linux 9.0!

Latest changes

PNY USB 6 in 1 card reader / Flash memory card reader	(2003-07-18)	show
Archos MiniHD 6GB / USB Cable Adaptor for MiniHD	(2003-07-18)	show
Creative Labs WebCam PRO Ex / model PD1050	(2003-07-11)	show
Anubis Electronic GmbH 40150 TYPHOON STREAM OPTICAL MOUSE / 5 buttons usb mouse + wheel	(2003-07-11)	show
Sitecom CN-300 / Sitecom CN-300 Multi-memory reader writer	(2003-07-09)	show

Latest additions

gamtec gamtec / gamtec	(2003-07-15)	show
Orite vc3110 / Digital camera 3.1 megapixel	(2003-07-11)	show
Lexar Media JumpDrive / 128MB pocket (keychain) hard drive.	(2003-07-09)	show
Systems Technology Pte Ltd Ranger RP USB2.0 Portable Hard Disk /	(2003-07-08)	show
Sony Sony DSC / Sony DSC-U20 digital still camera	(2003-07-08)	show

PCVC680 USB & Philips PCA645

Linux support for Philips USB webcams (and Linux only)

Welcome! On these pages you will find information over and downloads for the Linux driver for Philips USB webcams. This driver also supports some cameras from Askey, Creative Labs, Logitech, Samsung, Sotec and Visionite (and the list keeps growing).

Supported cameras:

- PCA645VC
- PCA646VC
- PCVC675K "Vesta"
- PCVC680K "Vesta Pro"
- PCVC690K "Vesta Scan"
- PCVC720K/40 "ToUCam XS"
- PCVC730K "ToUCam Fun"
- PCVC740K "ToUCam Pro"
- PCVC750K "ToUCam Scan"
- Askey VC010
- Creative Labs Webcam 5
- Creative Labs Webcam Pro Ex (soon)
- Logitech QuickCam 3000 Pro
- Logitech QuickCam 4000 Pro
- Logitech QuickCam Notebook Pro
- Logitech QuickCam Zoom
- Samsung MPC-C10
- Samsung MPC-C30
- Sotec Afina Eye
- Visionite VCS UM100
- Visionite VCS UC300

Look here for a table with the required PWC version.

NOT supported by this driver:

- PCA635VC (parallel version)
- PCVC665K "Vesta Fun" (look here)
- PCVC720K/20 "ToUCam XS" (look here)
- Logitech QuickCam Notebook (*without* Pro)
- Logitech Cordless
- new Creative Labs Webcam 5

Read the FAQ for the reason.

Confused about the double entries for the PCVC720K and Creative Labs Webcam 5?

News Status

Downloads Installation notes
F.A.Q. Support Bugtracker
Buying guide

Working software Webcams!
Example pictures API for programmers

Other software by me

Good news for **MacOS X** users! Work has started on a driver for MacOS X by Matthias Krauss. His work can be found here and at sourceforge.net. So your pleas may be (partially) answered. **Be careful**: older versions of the driver could damage your webcam, so make sure you get the latest one.

News

2003-07-14: Interesting news. Last week I received an email from Logitech regarding this driver... As you may have noticed, there are quite a few Logitech cameras supported by PWC, and this number is growing. In fact, they asked me if I was interested in providing support for new and upcoming Logitech cameras! Of course I said yes, and I think Logitech should be applauded for taking this pro-active step. Tonight I spent an hour on the phone with the manager of the Logitech video division, hashing out some details and explaining a bit about the Linux philosophy and driver model. There are some interesting things going to happen, so stay tuned!

2003-06-14: A new PWCX "super value plus" package is available for download. This package contains modules for a number of processors, kernel versions and compilers, so have fun!

2003-06-04: I'm back in the air! Well, on the Net, anyways. This has an advantage and a disadvantage. The good news is that it's now easier for me to post bugfixes, download kernel versions etc. The bad news is that it means I can also chat endlessly without having to watch the phone bill, which in turn means that I spend way too much time on IRC and not coding. Life without the Net isn't so bad after all...

2003-04-25: In two days, I'm going to move. This unfortunately also means the bugtracker will be offline, since it's running from my ADSL connection. However, I will be able to answer mail, so not all is lost :-)

2003-04-08: Rather unexpectedly, I'm going to move house. That means a lot of painting, wallpapering, running from shop to DYI market to counter, etc. You probably know the routine. That also means I won't have much time for PWC the next couple of weeks, plus it will take a long time to get ADSL up and running (why I can have a new phone line in 2 days but ADSL can take up to 12 weeks is beyond me... It's the same set of wires). I can dial in and receive mail, so I won't be completely out of touch. Just slow.

2003-03-06: While my suitcase is waiting to be filled, I made a quick fix to get the PVCV720K/40 and Creative Labs Webcam Pro Ex working on Linux 2.4.20.

2003-01-30: I've created a bug tracker database for PWC. Use it wisely!

2003-01-24: You gotta love those manufacturers... What's the case? Recently, new models of the Creative Webcam 5 and Philips PCVC720 have been manufactured. To add to the (already confusing) list of supported webcams, the new Creative cam is **not** supported by my driver (it uses a different chipset), but the PCVC 720K/40 **is**. You win some, you loose some... Unfortunately I don't know what distinguishes the new Webcam 5 from the old model, the PCVC 720K/40 is recognizable by it's /40 postfix.

2002-12-16: PWC 8.10 is out; there are 2 fixes (including the name of the QuickCam Notebook webcam) and one new ioctl (see the API). Patches has been sent; unfortunately it seems it didn't made it to 2.5.52 (it came out a few hours after my patches were applied). Since I have no idea when 2.4.21 will be out, and the PWC version in 2.4.20 is quite outdate, I've provided a package with the full source code (so no icky patches).

2002-12-07: It has been pointed out to me that there are two notebook webcams from Logitech: the QuickCam Notebook and QuickCam Notebook Pro (how original...). Only the Pro version works with the PWC driver; the other camera seems to be based on the QuickCam Express and may work with that driver. So watch out which version you're buying!

2002-11-28: PWC 8.9 is out, in kernel 2.5.50. It includes IDs for more Logitech cams, and finally all the others are in place as well (like the Visionites). Note: they drastically changed the module load process and I think PWCX will not load anymore, so I'll compile a new version and upload it soon.

2002-11-04: A **GCC 3.2** compiled version of PWCX is available for download. I've compiled and tested this module under 2.5.33, but should work on any GCC 3.2 compiled kernel. Have fun!

2002-10-29: RedHat, oh RedHat... Or rather, GCC in this case. Yes, I'm quite aware of the **GCC 3.2** compiler shipped with **RedHat 8.0**, but no, I haven't made a copy of PWCX for GCC 3.2 available yet. Mind you, 'available'. I did set up a test system with GCC 3.2 but haven't been able to get a working kernel with PWC and PWCX for various reasons. So please be patient while I'm sorting things out.

2002-10-05: The Logitech QuickCam Pro 4000 seems to be quite popular, I've had about 10 mails the past few days about this cam. Therefor, version 8.8 of PWC is available for download. Have fun!

2002-09-26:I've been away for a short vacation, a week of full of sun and doing nothing :-) (okay, I admit, I took my laptop with me). I'll be gone again this weekend, so answering your mail might take a few days extra.

Next, I have some bad and some good news for you. The bad news is that Philips doesn't seem to make any kind of webcam anymore, so if you want one, be quick. Shops in the US don't seem to carry them anymore, here in Europe they are still on sale. The good news is that the **Logitech Quickcam 4000** seems to be supported by this driver, so I'll bring out a patch soon.

2002-09-06: With the aid of Jacky Mallet, I've managed to set up cross-compiling for the **ARM** processor. Yay! I can tell you it wasn't easy; I ended up downloading a binary distribution, because all my attempts at cross-compiling GCC for something ARM-ish failed (and I'm not exactly a beginner....)

The big question now of course is: does it work? The PowerPC PWCX modules worked out of the box, let's hope the same applies to pwcx-sa1100.o.gz. This module is compiled with SA1100 turned on the kernel config; I'm not familiair with the various ARM processors, I hope it works on all ARM processors. So please let me know if it works!

2002-07-29: Small case of "whoops;": the PowerPC PWCX module was accidently compiled with module versioning on. A new versionless module is available for download.

2002-07-08: Am I still alive? Yes. Is work still being done on the driver? Yes, albeit slowly. There really isn't much to add, anyway. It's now mainly mainenance, keeping the driver alive in the 2.5.* kernel series. Recently, I've only been cleaning up the webcams section. Cams that appear to be gone have been removed; some links have been updated. Others seem to be standing still: they may get removed soon. If yours has disappeared, or want a new one added, drop me a line!

2002-04-30: You know, it's funny. I write only Linux software, this webcam driver and CamStream. Still, I'm getting an increasing amount of mail infected by the Klez virus, sometimes from people who, as far as I know, never mailed me before. Now I'm running KMail so it doesn't affect me, but it's growing to a point it starts to annoy me. So upgrade your virusscanner and, most of all, stop using using those bloody Microsoft products! (And oh, while we're at it: stop scanning my ADSL connection)

2002-03-26: What started out as a small item for the FAQ has grown into something more like an editorial... It's called Tainting the kernel.

2002-03-25: Small case of "oops": I compiled the PWCX 8.2 x86 module on my Pentium-II, and some AMD users reported an Oops due to illegal opcodes. I re-compiled the module and put a new PWCX 8.2 package on the site. On a related note: it's now been a month since the PowerPC decompressor module is out, and **nobody** reported anything about it, either success or failure. Did I really put all those hours of work in it for nothing?!? **Update:** At least two people confirmed it works. That's good :-)

Status

The 8.* modules have been in the kernel for more than 3 months now, and so far there have been only one medium-sized problem with memory allocation. Overall, the driver seems to be quite stable, and I am now working on enhancing the driver to support all functions that are available in the camera.

Of course, there are occasional problems with non-responsive cameras, hangs or cases of "simply not working". In a lot of cases it is a matter of not reading the documentation well, or other causes that have nothing to do with my modules (but due to the modular nature of USB support, is difficult to determine for someone who doesn't know the inner workings of the Linux USB subsystem).

The Video4Linux implementation is complete; with the inclusion of mmap() code, and the 'viewport' algorithm that allows non-supported image sizes, a lot of applications work, except those that can't handle the palette stuff (check out the Working stuff page). Unfortunately there are also applications written for TV grabber cards that will simply not work with webcams due to the way they handle video devices; supporting them will require hacking my driver, which is the wrong way around IMO.

But, with utilities like CamStream it's possible to generate a stream for several hours without any problems. If you do run into a problem, please read the FAQ first, then go on to the support page so you can reach me.

Releases of CamStream can be found here.

Thanks

A few words of thanks...

- Philips gratiously donated a PCVC680, a PCVC730 and a PCVC740 webcam for driver development, and kudos to their engineers which have to endure the stream of E-mails from me :)
- Thanks to Johannes Erdfelt, who wrote the initial CPiA webcam driver which was a great example to look at.
- To Randy Dunlap for providing me with a Zoom (CPiA based) webcam so I can keep development in parallel and to get the initial ISOC stuff working.
- The dozens of other developers who developed and tested the USB stuff under Linux.
- And finally, all the users who send me their bug reports. Sometimes its nothing but old news, other times it really helps to solve a problem

Disclaimer

Okay, just a small legal blurb to get my hiney covered.

This information and software is provided by me without any warranty of any kind. I can not be held responsible for fried computers, crashes, dates as a result of pictures taken by this camera or public embarrasment because you forgot to switch off your webcam program during a visit by aforementioned date :).

More seriously: because I don't charge for the drivers and don't make any money out of it, I cannot take any responsibility. There are a million different computers out there, and thus a million things that can go wrong.

Note: I am not an employee or contractor for Philips B.V., the Netherlands, or any of its business partners. These modules are provided on a volountarely basis, based on my own free time.

If you want to see how many people visit this webpage, click on this little icon:

2003-02-13 - Nemosoft Unv.

I am no longer the Maintainer for the Linux USB Scanner driver.

As of January 2003, Henning Meier-Geinitz has taken over the duties of maintianership.

Henning also has a www page available, too.

Try the Top Searches on the Web!

Search

Finance

Finance
Home Loans
Debt Consolidation
Mortgage
Loans

Free

Free
Free Coupons
Free Credit Report
Free Insurance Quotes
Free Casino Games

Beauty

Love
Personals
Romance
Matchmaking
Dating

Gambling

Online Gambling
Casinos
Blackjack
Poker
Card Games

Travel

Travel
Vacations
Cruises
Car Rental
Airline Tickets

Electronics

Electronics
Digital Camera
Pda
Dvd
Cellular Phones

Gifts

Gifts
Antiques
Wedding
Golf
Jewelry

Shopping

Shopping
Online Shopping
Video Games
Gift Baskets
Jewelry

Home

Home Business
Real Estate
Refinance
Home Loan
Work At Home

Office

Office Supplies
Marketing
Telemarketing
Phones
Printers

Computers

Web Design
Data Recovery
Video Projector
Domain Registration
Security

Entertainment

Dating
Gambling
Sports
Music
Games

Popular Searches

1. **Usb**
2. **Universal Serial Bus**
3. **Usb Adapter**
4. **Usb Cable**
5. **Usb Drive**
6. **Usb Network**
7. **Usb Serial**
8. **Electronics**
9. **Usb Interface**
10. **Usb Flash Drive**
11. **Digital Camera**
12. **Free**
13. **Software Consultants**
14. **Usb Bridge**
15. **Rs232**
16. **Printers**
17. **Usb Hub**
18. **Loader**
19. **Data Recovery**
20. **Fitness**

1799 Silacci Drive
Campbell, CA, 95008

Mail: Post Box 60280
Palo Alto, CA, 94306

Phone#: 650-465-4000
Fax#: 650-493-2200

Email:<salesinfo_AT_ActiveWireInc.com>

ActiveWire News

Specification

Download Software

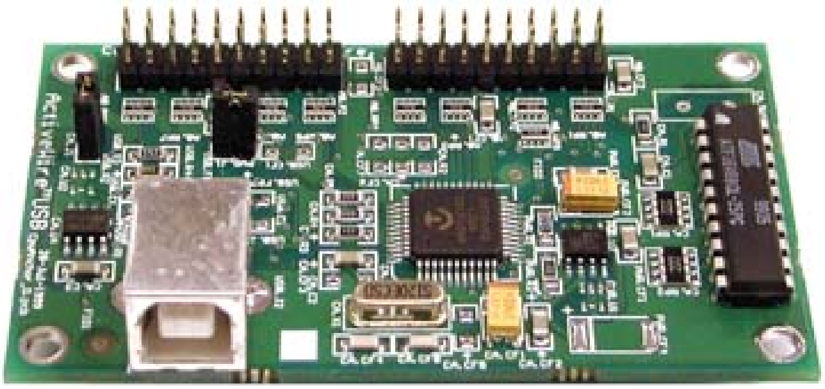
View TechNote

View Hardware Manual

View Software Manual

View Other Documentation

AW ActiveWire, Inc.



ActiveWire-USB . Price: $59.00
ActiveWire-USB + Cable . Price: $64.00
Buy Now

ActiveWire-USB with programmable I/O pins that can interface to anything. Control the pins from your browser by HTML/ javascript or VBasic or C. Macintosh, Win95/98/2K/ME/XP, Linux, FreeBSD, and LabView drivers.

We now have variety of Add-on boards to allow easy connection to other devices: Motors, Solenoids, Relays, LCD, etc...

Enter Online Store Now!!

Links to other sites

Design-In Program

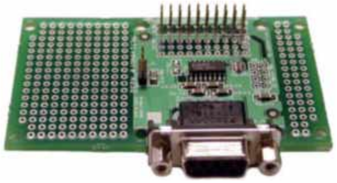
For volume products ActiveWire can provide:

- Pre-negotiated USB parts
- License object or source USB drivers for all platforms

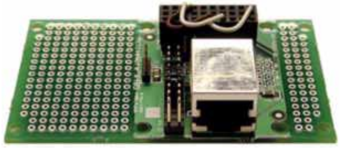
Custom Designs

ActiveWire offers custom design, development, and manufacturing.

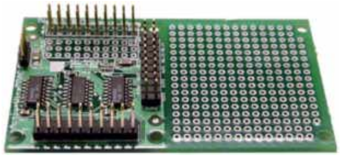
RS-232 Add-on board $32



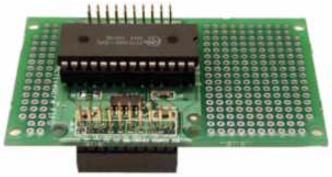
RS-485 Add-on board $39



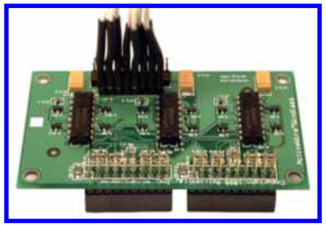
Digital to Analog Converter Add-on board $49



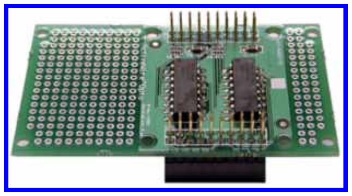
FIFO buffer Add-on board $49



Motor Control Add-on board $49



Optoisolator Add-on board $29



LCD character display module w/ Interconnection Add-on board $39

ActiveWire Add-on boards. Priced from $29.00 Buy Now!!

Thank You; Moto, Cisco, Intel, Compaq, AMD, Interval Research, Texas A&M, HP, Ericsson Radio, Qualcomm, National Semi, FAA, Pfizer, and all the individual customers.

another PICOSTAR company

ActiveWire ® is a registered trademark of PicoStar,LLC.

Products | Support | Sales

About Agere | News & Events | Investor Relations | Careers

Metro/Regional Transport

Enterprise/Metro Access

Client Solutions

Find Your Product

By Application

Part Number Listing

Products A-Z

Documentation Library

Search

GO

Site Map | Contact Us

Agere Systems Home > **Redirect**

The content of this page has moved to www.agere.com
You will be taken there automatically.
Please update your bookmarks

agere systems

Products | Support | Sales | About Agere | News & Events | Investor Relations | Careers

Find Your Product

- By Application
- Part Number Listing
- Products A-Z
- Documentation Library

Search

GO

Site Map | Contact Us

Choose Your Language

English GO

Straight to technical documentation >>

Agere Named #1 in Hard Disk Drive ICs and SONET/SDH Networking Equipment*

* IDC Market Analysis, May 2003 >>>

Festino™ System Card Solutions >>>

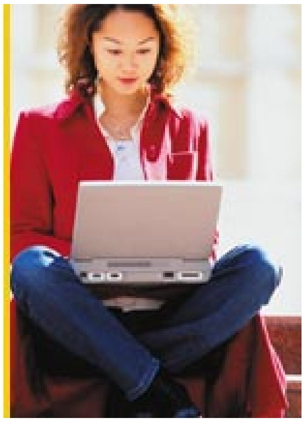
TrueStore™ Next generation storage solutions >>>

Download the Latest Driver

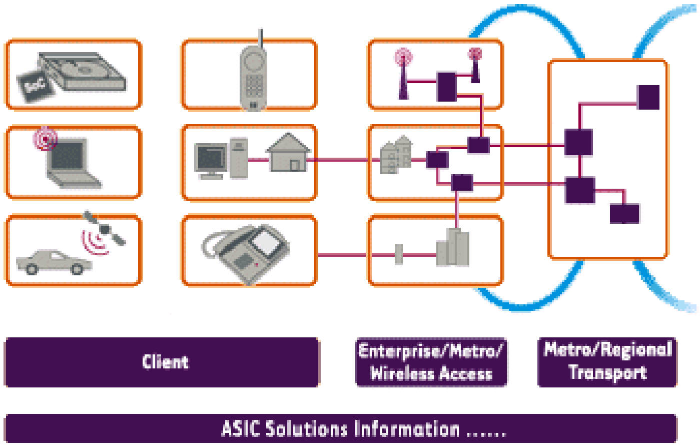
- Wireless LAN Drivers
- V.90 Modem Driver

High-Density Storage

Agere's market-leading data storage solutions for hard disk drives in PCs, laptops and servers enable faster, higher-capacity storage for data, graphic and video files.



Select Your Application Segment:


Client
Enterprise/Metro/ Wireless Access
Metro/Regional Transport
ASIC Solutions Information ......

News & Events

- Press Releases
- Tradeshows

More

Investor Information

- SEC Filings
- Stockholder Services
- Spin-Off Information

More

Careers

- Search Jobs

More

Latest News:

- **July 16, 2003**
  Agere Systems Announces GPRS Chip Set and Software That Significantly Improve Processing Power for Advanced Multimedia Mobile Phones
- **July 15, 2003**
  Agere Systems Announces Integrated Chip Set Pairing Wi-Fi™ and VoIP Technologies for...

Recent Articles:

- **EDN Magazine**
  When the package means as much as the chip
- **Electronic Design**
  Pre-Test SERDES To Smooth Integration In Fast ASICs
- **Inbound Logistics**
  Optimizing Your Supply Chain: A Model Approach

Archived Articles >>>

NetChip Technology, Inc.

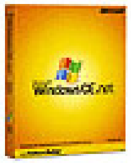
Welcome

- Home
- Company Information
- USB Products
- Development Kits
- Literature Library
- News and Events
- Links and Resources
- Reps and Distributors
- Contact Us

USB2.0

NEW High Speed USB 2.0 Performance Reports:
NET2280
NET2272

Linux Support (Click for driver source codes)

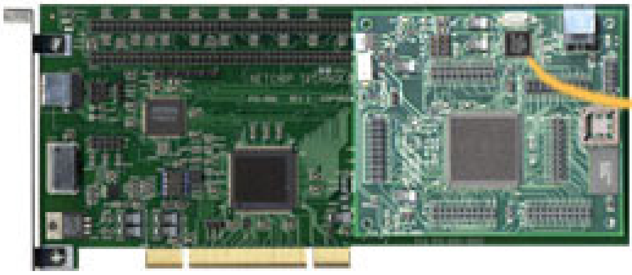


WindowsCE.net Support

FREE Request for USB 2.0 Solution CD

Last Updated: 07/07/2003

Introducing *NET2272* and the new *PCI-RDK*



Win a FREE USB 2.0 Device Development Kit!

(Click here for details)

NET2280RDK
NET2280 development kit (includes PCI host support)

NET2280EVB-SW
Includes Evelyn PCI board and development software

Drivers Tech Support Contact Us Site Map

about us support news investor relations

OPTi

LCD Panel Controllers

PCI Bridge Solutions

USB/1394 Solutions

Mobile Solutions

USB/1394 SOLUTIONS

FireLink™ Now available in two exciting flavors, with two more on the way!

FireLink 82C861 - the Industry Standard USB Controller

USB

OPTi offers the world's "most compatible" USB solution. Introduced in 1997, the 82C861 controller was the first practical OHCI solution available, and continues to outsell the competition due to its high quality and low price. Two ports, 12Mbps bandwidth, and your choice of PQFP or LQFP 100-pin packages make the FireLink a perfect design solution for adding USB capabilities to motherboards, in embedded applications, on a PCI card or through CardBus.

FireLink 82C861 Controller features:

- Windows 98/Me/CE/2000 support
- Apple Mac OS support
- OpenHCI
- Dual USB ports
- 1.5 Mbps and 12Mbps data rate
- USB Rev 1.1 and PCI Rev 2.1 compliant
- Same device operates at 3.3V or 5V
- Managed power consumption
- Clock stop capable
- Pin compatible with CMD, Lucent parts
- 100-pin PQFP/LQFP

OPTi Part Numbers:

- QP0086110XUE-002 (PQFP package, 20x30 pin)
- QT0086110XUE-002 (LQFP package, 25x25 pin)

Please call for price quotes.

FireLink USB 82C862 - Supercharged USB Solution

With the 82C862 controller, OPTi integrates two 82C861 controllers into a single 100-pin package, offering 4 ports and 24Mbps bandwidth -- all for a price in the range of competitive single-controller solutions.

FireLink USB 82C862 Controller features:

- Windows 98/Me/CE/2000 support
- Apple Mac OS support
- OpenHCI
- Dual USB controllers, 12Mbps each
- Four USB ports (two port version: 82C863)
- 1.5 Mbps and 12Mbps data rate
- USB Rev 1.1, PCI Rev 2.2 compliant
- 3.3V operation, 5V-tolerant I/O
- PCI power management
- Ultra-low power consumption
- 100-pin LQFP

OPTi Part Number:

- QT0086202XME-002 (LQFP package, 25x25 pin, four ports)
- QT0086302XME-002 (LQFP package, 25x25 pin, two ports)

Please call for price quotes.

Driver Support

OPTi USB controllers are supported directly by built-in drivers in the Windows 98, Windows Me, and Windows 2000 operating systems, as well as by the Apple Mac OS (click here for Mac drivers from Apple).

For information about a USB host stack and broad library of class drivers for non-Windows operating systems, see OPTi's software partner, SoftConnex Technologies, Inc., which provides USB software solutions to the embedded market. The SoftConnex USB product line includes drivers for VxWorks, Nucleus, PowerTV, QNX, Linux, MS-DOS and other operating systems.

SoftConnex
CONNECTING PERIPHERALS TO INTELLIGENT DEVICES

FireLink Turnkey Manufacturing Package

In an effort to speed up time to market, OPTi is pleased to offer a FireLink Turnkey Manufacturing Package. This turnkey package is aimed at board manufactures wishing to capture the USB upgrade market. When end users upgrade to Windows 98, they will want to add USB capability to their non-USB system. The FireLink Turnkey Manufacturing Package kit includes the following:

- Two layer PCI-to-USB add-in card

- OrCAD Schematics
- Bill of Materials
- Gerber Files
- Job File
- Databook

Turnkey packages are offered at no charge. You can download them from the selections below. Please contact OPTi for additional information.

82C861

Data Book

File Name	Date	Size	Description
usb1.pdf	05/15/98	282KB	FireLink USB 82C861 PCI-to-USB Bus Bridge Data Book
ad013_13.pdf	10/03/00	23KB	FireLink USB 82C861 Data Book Addendum
an069_10.pdf	03/05/97	24KB	FireLink USB Board Level Design Considerations Application Note
861schem.pdf	03/12/99	53KB	FireLink USB Reference Schematics: Use these schematics as a guideline when designing with OPTi's FireLink 82C861.

2-Port USB Board - OPTi Order Number 800-0162-000

File Name	Date	Size	Description
2p861tmp.zip	5/26/98	141KB	2-Port USB Turnkey Manufacturing Package (OrCAD 4 schematics, Gerber files, job files)
2p861oc9.zip	3/2/01	37KB	2-Port USB TMP Schematics (OrCAD 9)
ce0162.zip	04/16/01	164KB	2-Port USB CE Statement of Compliance

USB CardBus Design

File Name	Date	Size	Description
cb861ocd	12/15/99	10KB	USB CardBus Schematics (OrCAD)
cb861gbr.zip	12/15/99	76KB	USB CardBus Gerber Files
firelinkcb.zip	02/18/00	81KB	USB CardBus Schematics (PDF)

82C862/82C863

USB Controller Data Book

File Name	Date	Size	Description
db030_30.pdf	5/18/01	484KB	FireLink USB 82C862/863 Controller Data Book
tb042_00.pdf	6/05/01	492KB	FireLink USB 82C862/863 Product Brief
pa059_00.pdf	1/9/01	10KB	Product Alert - Clock Issue
pa060_00.pdf	3/13/01	9KB	Product Alert - Compatibility Issue

4-Port USB Board - OPTi Order Number 800-0190-000R1.0

File Name	Date	Size	Description
4p862ocd.zip	1/9/01	61KB	4-Port USB Schematics (OrCAD)
4p862gbr.zip	1/4/01	204KB	4-Port USB Gerber Files
4p862job.zip	1/9/01	101KB	4-Port USB Job Files
4p862bom.zip	1/9/01	6KB	4-Port USB Bill of Materials
fcc0190.tif	11/29/00	10KB	4-Port USB FCC Statement of Compliance
ce0190.zip	03/13/01	163KB	4-Port USB CE Statement of Compliance

Universal 2-port USB Board OPTi Order Number 800-0300-000

File Name	Date	Size	Description
2p862ocd.zip	12/15/99	54KB	2-Port USB Schematics (OrCAD)
2p862gbr.zip	12/15/99	157KB	2-Port Gerber Files

1394

FireLink 1394 PCI Combo 1394/USB Board

File Name	Date	Size	Description
comboocd.zip	12/15/99	78KB	PCI Combo Card Schematics (OrCAD)
combogbr.zip	12/15/99	217KB	PCI Combo Card Gerber Files
cg0189_01.pdf	12/15/99	449KB	PCI Combo Card Configuration Guide
ug029_01.pdf	12/15/99	16KB	PCI Combo Card User's Guide

You are being redirected to http://www.ti.com/sc/docs/products/msp/intrface/usb/overview.htm

You will be redirected to
http://focus.ti.com/analog/docs/articles.tsp?familyId=361&templateId=5&path=templatedata/cm/brc/data/20020709_usb_landing_page&articleType=brc

Interface : Universal Serial Bus (USB)

TI USB Solutions	Transient Voltage Supressors	USB Solutions Guide
USB Home	Power Management	1394 Home
Hub Controllers	USB Block Diagrams	UART Home
Peripheral Controllers	Evaluation Modules	Connectivity Home
Streaming Audio	Developer Network	Connectivity Press Releases
USB Resources		

USB has

greatly simplified the lives of PC users by combining multiple existing interfaces into a single, easy-to-use connector. USB's plug-and-play capability ends the formerly complex process of adding system peripherals.

USB offers three speeds: low-speed (1.5 Mbps), full-speed (12 Mbps, or USB 1.1), and high-speed (480 Mbps, or USB 2.0). All three offer both asynchronous and isochronous (real-time) data transmission over a simple and inexpensive 4-wire cable. This meets the requirements of many peripherals, including keyboards, mice, printers, speakers, scanners, external storage devices and digital still cameras.

TI USB Solutions

TI's high-performance portfolio includes fully compliant USB hub controllers, peripheral devices, power management products and streaming audio devices. In addition to a broad range of silicon solutions, TI has the support tools, software, documentation, and systems expertise

Related Products

- Amplifiers and Comparators
- Data Converters
- Interface
- MSP430 Ultra-Low-Power MCU
- Power Management
- Audio
- Clocks & Timers
- Controls & Monitoring
- RF
- Wireless & Telecom
- Video & Imaging
- Special Functions

Support

- KnowledgeBase
- Email Tech Support
- Training

Applications

- All TI Applications
- Audio
- Avionics
- Data Communications
- Digital Speakers
- Digital Still Cameras
- Electronic Countermeasures

to help simplify design and speed your time to market.

Learn more about TI's hubs, peripherals, power management devices, and audio products.

Products | **Applications** | **Support** | **Site Map**