The website of Frank Zhao, electrical engineer and DIY hobbyist

Tutorial about USB HID Report Descriptors

This page is from my old website, and it is sort of popular, so I've moved it here.

A USB HID report descriptor is one of the descriptors that a USB host can request from a USB device. HID devices send data to the host using reports, and the descriptor tells the host how to interpret the data. I will try to show you how to write one of these descriptors.

First, go to this page http://www.usb.org/developers/hidpage/ and find the document titled "Device Class Definition for HID". What I will be talking about is essentially paraphrasing the important sections of that document.

Second, go get the HID descriptor tool from the same page. You'll want to play with it as you go through this tutorial. It is an absolute headache to write the HID report descriptors manually (converting between binary and hex and looking up the meanings of the numbers) so this tool is essential.

What is a USB HID report descriptor?

> *The HID protocol makes implementation of devices very simple. Devices define their data packets and then present a "HID descriptor" to the host. The HID descriptor is a hard coded array of bytes that describe the device's data packets. This includes: how many packets the device supports, how large are the packets, and the purpose of each byte and bit in the packet. For example, a keyboard with a calculator program button can tell the host that the button's pressed/released state is stored as the 2nd bit in the 6th byte in data packet number 4 (note: these locations are only illustrative and are device specific). The device typically stores the HID descriptor in ROM and does not need to intrinsically understand or parse the HID descriptor. Some mouse and keyboard hardware in the market today are implemented using only an 8-bit CPU.*

– Wikipedia on Human Interface Device

I'm going to try teaching you about USB HID report descriptors by walking you through writing a few.

For a simple starting point, let us make a standard mouse. Just three buttons, and movement on the X and Y axis. So we want to send data regarding the buttons and movement. It takes one bit to represent each button, and one byte to represent the movement on one axis as a signed integer. So we can say that we want the data structure to look something like this

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Useless	Useless	Useless	Useless	Useless	Left Button	Middle Button	Right Button
Byte 1	X Axis Relative Movement as Signed Integer							
Byte 2	Y Axis Relative Movement as Signed Integer							

And then we can say our data structure in C looks like

```
struct mouse_report_t
{
    uint8_t buttons;
    int8_t x;
    int8_t y;
}
```

So now in our descriptor, our first item must describe buttons, three of them

```
USAGE_PAGE (Button)
USAGE_MINIMUM (Button 1)
USAGE_MAXIMUM (Button 3)
```

each button status is represented by a bit, 0 or 1

```
LOGICAL_MINIMUM (0)
LOGICAL_MAXIMUM (1)
```

there are three of these bits

```
REPORT_COUNT (3)
REPORT_SIZE (1)
```

send this variable data to the computer

```
INPUT (Data,Var,Abs)
```

and the final result looks like

```
USAGE_PAGE (Button)
USAGE_MINIMUM (Button 1)
USAGE_MAXIMUM (Button 3)
LOGICAL_MINIMUM (0)
LOGICAL_MAXIMUM (1)
REPORT_COUNT (3)
REPORT_SIZE (1)
INPUT (Data,Var,Abs)
```

that will represent the buttons

but what about the five useless padding bits?

```
REPORT_COUNT (1)
REPORT_SIZE (5)
INPUT (Cnst,Var,Abs)
```

now we make the X axis movement

```
USAGE_PAGE (Generic Desktop)
USAGE (X)
```

we want it to be a signed integer that takes one byte, so it has a value between -127 and +127 (actually -128 and +127, but I want to keep things even)

```
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
```

we want it to take an entire byte which is 8 bits

```
REPORT_SIZE (8)
REPORT_COUNT (1)
```

and send it to the computer as a variable relative coordinate

```
INPUT (Data,Var,Rel)
```

you end up with something like this to represent the X axis movement

```
USAGE_PAGE (Generic Desktop)
USAGE (X)
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (1)
INPUT (Data,Var,Rel)
```

How about Y axis? You can try

```
USAGE_PAGE (Generic Desktop)
USAGE (X)
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (1)
INPUT (Data,Var,Rel)
USAGE_PAGE (Generic Desktop)
USAGE (Y)
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (1)
INPUT (Data,Var,Rel)
```

Which will work, but to save memory, we can do this instead

```
USAGE_PAGE (Generic Desktop)
USAGE (X)
USAGE (Y)
LOGICAL_MINIMUM (-127)
```

```
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (2)
INPUT (Data,Var,Rel)
```

So all your data will end up looking like

```
USAGE_PAGE (Button)
USAGE_MINIMUM (Button 1)
USAGE_MAXIMUM (Button 3)
LOGICAL_MINIMUM (0)
LOGICAL_MAXIMUM (1)
REPORT_COUNT (3)
REPORT_SIZE (1)
INPUT (Data,Var,Abs)
REPORT_COUNT (1)
REPORT_SIZE (5)
INPUT (Cnst,Var,Abs)
USAGE_PAGE (Generic Desktop)
USAGE (X)
USAGE (Y)
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (2)
INPUT (Data,Var,Rel)
```

Ah but we are not done, in order to make the computer know that this is a mouse, we do

```
USAGE_PAGE (Generic Desktop)
USAGE (Mouse)
COLLECTION (Application)
    USAGE (Pointer)
    COLLECTION (Physical)

    ... What we wrote already goes here

    END COLLECTION
END COLLECTION
```

So in the end, this is the USB HID report descriptor for a standard mouse

```
0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
0x09, 0x02,                    // USAGE (Mouse)
0xa1, 0x01,                    // COLLECTION (Application)
0x09, 0x01,                    //   USAGE (Pointer)
0xa1, 0x00,                    //   COLLECTION (Physical)
0x05, 0x09,                    //     USAGE_PAGE (Button)
0x19, 0x01,                    //     USAGE_MINIMUM (Button 1)
0x29, 0x03,                    //     USAGE_MAXIMUM (Button 3)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x25, 0x01,                    //     LOGICAL_MAXIMUM (1)
0x95, 0x03,                    //     REPORT_COUNT (3)
0x75, 0x01,                    //     REPORT_SIZE (1)
0x81, 0x02,                    //     INPUT (Data,Var,Abs)
0x95, 0x01,                    //     REPORT_COUNT (1)
0x75, 0x05,                    //     REPORT_SIZE (5)
0x81, 0x03,                    //     INPUT (Cnst,Var,Abs)
0x05, 0x01,                    //     USAGE_PAGE (Generic Desktop)
0x09, 0x30,                    //     USAGE (X)
0x09, 0x31,                    //     USAGE (Y)
0x15, 0x81,                    //     LOGICAL_MINIMUM (-127)
0x25, 0x7f,                    //     LOGICAL_MAXIMUM (127)
0x75, 0x08,                    //     REPORT_SIZE (8)
0x95, 0x02,                    //     REPORT_COUNT (2)
0x81, 0x06,                    //     INPUT (Data,Var,Rel)
```

```
25  0xc0,                    //   END_COLLECTION
26  0xc0                     // END_COLLECTION
```

This is actually the example descriptor provided with the USB HID documentation, and you can also find this as an example provided with the HID tool.

Cool, at this point, you will have encountered some concepts that you may have questions about, you should research the following:

Usage Pages

There's one thing that I think isn't explained well in the documentation, USAGE, USAGE_PAGE, USAGE_MINIMUM and USAGE_MAXIMUM. In a descriptor, you first set a USAGE_PAGE, and certain USAGEs are available. In the mouse example, USAGE_PAGE (Generic Desktop) allowed you to use USAGE (Mouse), and when the usage page was changed to USAGE_PAGE (Button), then the USAGE_MINIMUM and USAGE_MAXIMUM allowed you to specify the buttons, and before you can use USAGE (X) and USAGE (Y), the usage page was changed back to USAGE_PAGE (Generic Desktop). The usage page is like a namespace, changing the usage page affects what "usages" are available. Read the documentation called " HID Usage Tables" for more info.

Collections

Read the documentation about the official proper use of collections. In my own words, collections can be used to organize your data, for example, a keyboard may have a built-in touchpad, then the data for the keyboard should be kept in one application collection while the touchpad data is kept in another. We can assign an "Report ID" to each collection, which I will show you later.

Hey here's something you can do, by turning "USAGE (Mouse)" into "USAGE (Gamepad)", you make the computer think that it's a game pad with one joystick and 3 buttons. How about converting a Playstation 2 controller into a USB gamepad? The controller has 16 buttons and two thumb sticks, so we want the data to look like

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Button	Button	Button	Button	Button	Button	Button	Button
Byte 1	Button	Button	Button	Button	Button	Button	Button	Button
Byte 2	Left X Axis as Signed Integer							
Byte 3	Left Y Axis as Signed Integer							
Byte 4	Right X Axis as Signed Integer							
Byte 5	Right Y Axis as Signed Integer							

So our data structure looks like

```
struct gamepad_report_t
{
    uint16_t buttons;
    int8_t left_x;
    int8_t left_y;
    int8_t right_x;
    int8_t right_y;
}
```

We make the computer understand that it's a game pad

```
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)

    ...

    END COLLECTION
END COLLECTION
```

for the buttons

```
USAGE_PAGE (Button)
USAGE_MINIMUM (Button 1)
USAGE_MAXIMUM (Button 16)
LOGICAL_MINIMUM (0)
LOGICAL_MAXIMUM (1)
REPORT_COUNT (16)
REPORT_SIZE (1)
INPUT (Data,Var,Abs)
```

for the four thumb stick axis

```
USAGE_PAGE (Generic Desktop)
USAGE (X)
USAGE (Y)
USAGE (Z)
USAGE (Rx)
LOGICAL_MINIMUM (-127)
LOGICAL_MAXIMUM (127)
REPORT_SIZE (8)
REPORT_COUNT (4)
INPUT (Data,Var,Abs)
```

NOTE: Z is used to represent the right stick's X axis, Rx is used to represent the right stick's Y axis. This doesn't make sense but this is how most existing USB game pads work. I have tested this using Battlefield Bad Company 2, it works.

NOTE: Use "absolute" for something like joysticks, but "relative" for things like mouse.

So now you end up with

```
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
```

```
COLLECTION (Application)
    COLLECTION (Physical)
        USAGE_PAGE (Button)
        USAGE_MINIMUM (Button 1)
        USAGE_MAXIMUM (Button 16)
        LOGICAL_MINIMUM (0)
        LOGICAL_MAXIMUM (1)
        REPORT_COUNT (16)
        REPORT_SIZE (1)
        INPUT (Data,Var,Abs)
        USAGE_PAGE (Generic Desktop)
        USAGE (X)
        USAGE (Y)
        USAGE (Z)
        USAGE (Rx)
        LOGICAL_MINIMUM (-127)
        LOGICAL_MAXIMUM (127)
        REPORT_SIZE (8)
        REPORT_COUNT (4)
        INPUT (Data,Var,Abs)
    END COLLECTION
END COLLECTION
```

Hey how about two players? Here's where collections get handy

```
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (1)
        ...
    END COLLECTION
END COLLECTION
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (2)
        ...
    END COLLECTION
END COLLECTION
```

fill in the data areas and you end up with

```
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (1)
        USAGE_PAGE (Button)
        USAGE_MINIMUM (Button 1)
        USAGE_MAXIMUM (Button 16)
        LOGICAL_MINIMUM (0)
        LOGICAL_MAXIMUM (1)
        REPORT_COUNT (16)
        REPORT_SIZE (1)
        INPUT (Data,Var,Abs)
        USAGE_PAGE (Generic Desktop)
        USAGE (X)
        USAGE (Y)
        USAGE (Z)
        USAGE (Rx)
        LOGICAL_MINIMUM (-127)
        LOGICAL_MAXIMUM (127)
        REPORT_SIZE (8)
        REPORT_COUNT (4)
        INPUT (Data,Var,Abs)
```

```
    END COLLECTION
END COLLECTION
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (2)
        USAGE_PAGE (Button)
        USAGE_MINIMUM (Button 1)
        USAGE_MAXIMUM (Button 16)
        LOGICAL_MINIMUM (0)
        LOGICAL_MAXIMUM (1)
        REPORT_COUNT (16)
        REPORT_SIZE (1)
        INPUT (Data,Var,Abs)
        USAGE_PAGE (Generic Desktop)
        USAGE (X)
        USAGE (Y)
        USAGE (Z)
        USAGE (Rx)
        LOGICAL_MINIMUM (-127)
        LOGICAL_MAXIMUM (127)
        REPORT_SIZE (8)
        REPORT_COUNT (4)
        INPUT (Data,Var,Abs)
    END COLLECTION
END COLLECTION
```

This is really important: You must change your data structure to include the report ID

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Report ID							
Byte 1	Button	Button	Button	Button	Button	Button	Button	Button
Byte 2	Button	Button	Button	Button	Button	Button	Button	Button
Byte 3	Left X Axis as Signed Integer							
Byte 4	Left Y Axis as Signed Integer							
Byte 5	Right X Axis as Signed Integer							
Byte 6	Right Y Axis as Signed Integer							

```
struct multiplayer_gamepad_report_t
{
    uint8_t report_id;
    uint16_t buttons;
    int8_t left_x;
    int8_t left_y;
    int8_t right_x;
    int8_t right_y;
}
```

You must manually set the report ID before you send the data to the computer in order for the computer to understand which player the data belongs to.

```
multiplayer_gamepad_report_t player1_report;
```

```
multiplayer_gamepad_report_t player2_report;
player1_report.report_id = 1;
player2_report.report_id = 2;
```

You can also use collections and report IDs to make composite devices. So far I've shown you the keyboard, mouse, and gamepad. Here's something that describes a composite device that is a keyboard, mouse, and two player game pad.

```
USAGE_PAGE (Generic Desktop)
USAGE (Keyboard)
COLLECTION (Application)
    REPORT_ID (1)
    ...
END COLLECTION
USAGE_PAGE (Generic Desktop)
USAGE (Mouse)
COLLECTION (Application)
    USAGE (Pointer)
    COLLECTION (Physical)
        REPORT_ID (2)
        ...
    END COLLECTION
END COLLECTION
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (3)
        ...
    END COLLECTION
END COLLECTION
USAGE_PAGE (Generic Desktop)
USAGE (Game Pad)
COLLECTION (Application)
    COLLECTION (Physical)
        REPORT_ID (4)
        ...
    END COLLECTION
END COLLECTION
```

and of course, your data structures with the added report ID.

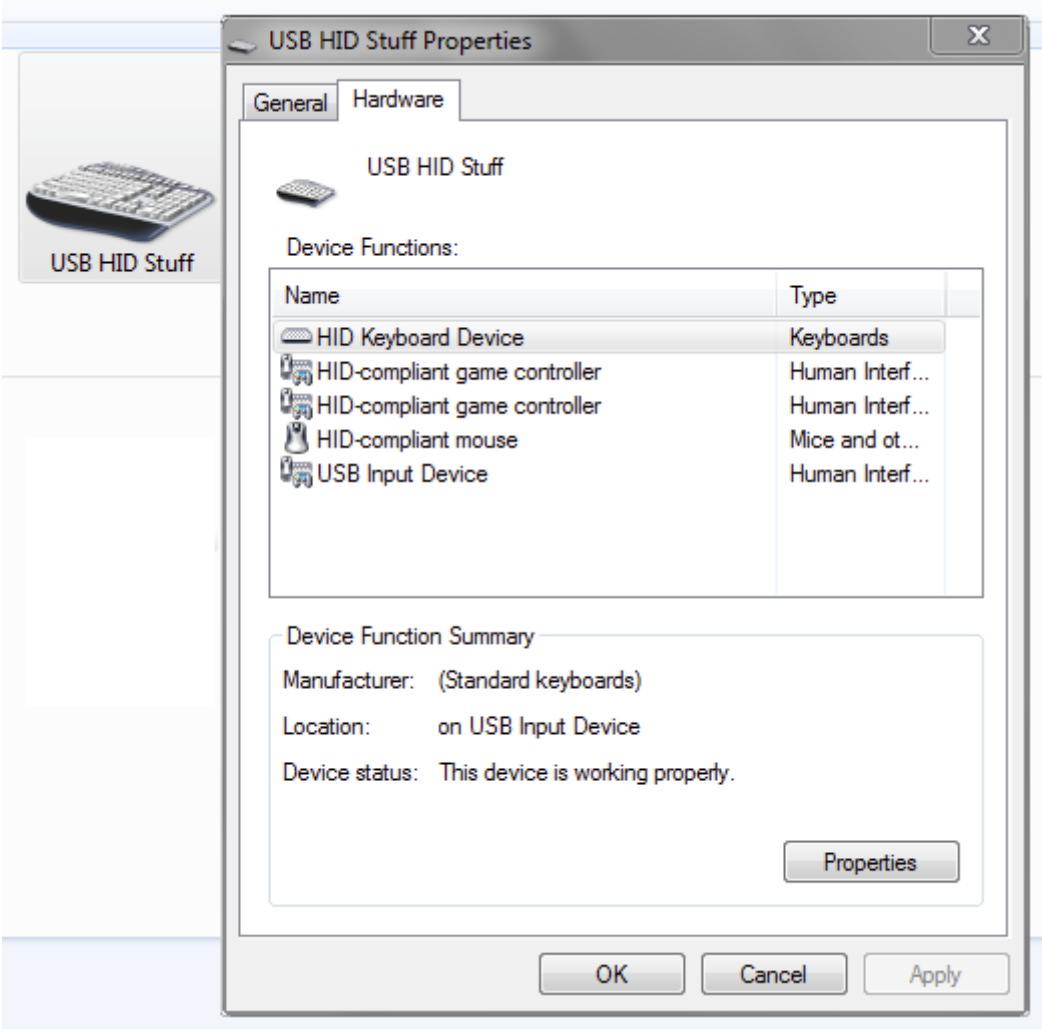
```
struct keyboard_report_t
{
    uint8_t report_id;
    uint8_t modifier;
    uint8_t reserved;
    uint8_t keycode[6];
}

struct mouse_report_t
{
    uint8_t report_id;
    uint8_t buttons;
    int8_t x;
    int8_t y;
}

struct gamepad_report_t
{
    uint8_t report_id;
    uint16_t buttons;
    int8_t left_x;
```

```
22      int8_t left_y;
23      int8_t right_x;
24      int8_t right_y;
25  }
```

But since we changed the data structure, your device no-longer supports boot protocol, and hence you will not need to define a protocol. So make sure you change usbconfig.h accordingly.

Want to see this in action? Load up this example project into USnooBie and let Windows's "Devices and Printers" show you!

Example Project Files



This entry was posted in Tutorial and tagged usb on January 1, 2013 [http://eleccelerator.com/tutorial-about-usb-hid-report-descriptors/] .

---

96 thoughts on "Tutorial about USB HID Report Descriptors"

Happy Fellow
December 2, 2013 at 7:17 PM

God bless you

CT
December 25, 2013 at 3:20 PM

Awesome tutorial…Thanks. CT

Sharan
January 2, 2014 at 5:44 AM

Really helpful. Thanks a lot!

SD Meijer
January 3, 2014 at 3:20 PM

Can you help me with a combo device (keyboard & mouse)?
I have two usbHidReportDescriptors.
The first one works under Windows, but under Linux everything works except the left mouse button.
The second one works under Linux, but under Windows the mouse doesn't work at all.
I posted a message on the V-Usb forum, maybe you can help?
http://forums.obdev.at/viewtopic.php?f=8&t=8785

Admin Post author
January 3, 2014 at 6:25 PM

Did you try making it look like a standard mouse report descriptor?

SD Meijer
January 3, 2014 at 7:01 PM

Yes, I did. I used your example. But that one doesn't work under Windows (at least not in combination with a keyboard). See the forum post at obdev.at.

SD Meijer
January 3, 2014 at 7:02 PM

I tested a lot. It looks like that whatever I set as buttonmask, it always is a right mouse click (under Linux). Under Windows it works without problems.
P.S. This is not your descriptor, but the other one. Because your descriptor doesn't work under Windows (7), although my Device Manager recognizes a mouse and says it's working correct.

SD Meijer
January 3, 2014 at 8:06 PM

Tried your descriptor again (without keyboard) and that worked.
Then I'd added the keyboard part and now both works!
I don't know what happened, but I'm happy with the outcome.

G Snyder
January 12, 2014 at 10:09 PM

Thanks so much for this outline. I'm sure I would have spent days pawing uncomprehendingly through hundreds of pages of specification without it. I just wanted to set up a simple Play/Pause control for an iPhone via BTLE, and this was exactly what I needed.

locodog
January 13, 2014 at 1:42 PM

Hello and thanks for the great outline, I'm trying to make a custom generic HID device with 128 single bit buttons and 48 10bit ADC analogue inputs, I'm a bit stumped as to what usage type to use?

All the source I have been using is for a gamepad, but I suspect I'll have to change this.

(I'm making a HID dj software controller NOT MIDI, as far as the software is concerned HID is HID)

Admin Post author
February 12, 2014 at 4:02 PM

If you don't see an appropriate "usage" or "usage page" in the HID specifications, then just use "vendor" or "gamepad". If you write your own DJ software, it doesn't matter as long as the data is accessible.

Picking the right "usage" is only important when you are dealing with other people's software, if you have total control over the software, it doesn't matter, as long as the data gets there. If you can get the data, you can use it.

locodog
February 28, 2014 at 9:43 PM

Many thanks for the reply, I am pleased to say I've got my project reporting 64 Bytes accurately, next thing is to work out 2 page reports (and I'm sure the answer is here, I just need to read and try the examples) after that I'll look at return packets (from the pc to the usb device)

Many thanks for the tutorial.

The Sparkbuzz
January 19, 2014 at 1:48 PM

Thanks for this article. It's been a great help. Working through the USB documentation is a horrible nightmare...

Jumbo
February 9, 2014 at 12:30 AM

Hi, thanks for all the info..
I also found it hard to find good info on HID. I have noticed it is possible to create a seemingly valid report description in the Report Descriptor Tool, but windows will not enumerate it.
I need to do bi-directional data transfer between stm32-f105 and windows host.

First I tried to customize the casual mouse report description, by setting the report size to 64 .. but my device would not enumerate under windows. ..Then I found this on internet:

```
HID_UsagePageVendor(0x00),
HID_Usage(0x01),
HID_Collection(HID_Application),

HID_LogicalMin(0),
HID_LogicalMax(255),
HID_ReportSize(8),

HID_ReportCount(64),
HID_Usage(0x01),
HID_Input(HID_Data | HID_Variable | HID_Absolute),

HID_ReportCount(64),
HID_Usage(0x01),
HID_Output(HID_Data | HID_Variable | HID_Absolute),
HID_EndCollection
```

It enumerates, but does it look correct to you? (I get weird behaviour on my device, so I would like to know if my report description is good before I start searching for bugs in the code – I already spend 3 days trying to get it to work :-/)

Admin Post author
February 9, 2014 at 4:51 AM

I've seen the same thing before, maybe from LUFA or Teensy's website or somewhere, I don't remember exactly. It usually works but doesn't do anything without a host application.

Compare it against your own descriptor.

Jumbo
February 9, 2014 at 11:30 AM

Lots of info on LUFA, I will have a look, thanks for sharing!

locodog
March 2, 2014 at 7:02 PM

If I wanted to send more than 64 bytes from one device, what would I do, just add a REPORT_ID (1) after the collection, then REPORT_ID (2) before the end collection and a then repeat what was in the collection when it was a 1 page, 64 byte report?

```
static const uint8_t PROGMEM gamepad_hid_report_desc[] = {
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x05, // USAGE (Gamepad)
0xa1, 0x01, // COLLECTION (Application)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x35, 0x00, // PHYSICAL_MINIMUM (0)
0x45, 0x01, // PHYSICAL_MAXIMUM (1)
0x75, 0x01, // REPORT_SIZE (1)
0x95, 0x80, // REPORT_COUNT (128)
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x80, // USAGE_MAXIMUM (Button 128)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x26, 0xff, 0x00, // LOGICAL_MAXIMUM (255)
0x46, 0xff, 0x00, // PHYSICAL_MAXIMUM (255)
0x09, 0x30, // USAGE (X)
0x09, 0x31, // USAGE (Y)
0x09, 0x32, // USAGE (Z)
0x09, 0x35, // USAGE (Rz)
0x09, 0x36, // Usage (Slider)
.....43 more sliders.
0x09, 0x36, // Usage (Slider)
0x75, 0x08, // REPORT_SIZE (8)
0x95, 0x30, // REPORT_COUNT (48)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0 // END_COLLECTION
};
```

Admin Post author
March 2, 2014 at 10:29 PM

You should read this page http://www.beyondlogic.org/usbnutshell/usb4.shtml where it talks about the size limits of the different endpoint types.

Basically, you are not supposed to design something that needs more than 64 bytes if you plan on using full-speed specs. There is no “work around” unless you use high-speed specs (meaning you need high-speed ca-

pable hardware, typically a ULPI PHY). You can try to split your data up in the application layer though, but that wouldn't work well if you want generic HID to work. There is no easy answer, sorry.

Neeraj
March 4, 2014 at 8:50 PM

Extremely awesomely helpful tutorial. God bless you!

locodog
March 10, 2014 at 11:55 AM

Hi again Frank, further to >64 bytes output, I believe I may have used the wrong language and caused confusion, My device sents 64 bytes of data to the pc, I expanded the HID report descriptor to include an extra 32 * 16 bit sliders, (also declared and defined these inputs in other parts of the code where needed)
My device now sends a HID report of 128 bytes of data, but the problem is that the new 64 bytes is an exact copy of the first 64 bytes (byte 64 is a mirror of byte zero, etc)

Does this sound like a Report descriptor error and can you recommend anything I should check?

Admin Post author
March 10, 2014 at 5:25 PM

If the second chunk is a copy of the first chunk, then it's a programming error, not a descriptor error, descriptor is not code. Make sure the first chunk is sent successfully and then fill in the memory for the second chunk.

I don't think sending 128 bytes in two chunks is a good idea for HID if you want a generic driver to take care of it.

Gavin
March 16, 2014 at 6:07 PM

Hi this is helping a lot with understanding USB protocols but i am still struggling with the hid descriptor tool and 10 bit axis values for X,Y,Z and slider, could you help.
Thanks, Gavin.

Admin Post author
March 16, 2014 at 6:19 PM

What have you tried? What did you expect to see happen? What happened instead?

Gavin
March 16, 2014 at 8:43 PM

Sorry didnt even give a good reply, i have the demo code mentioned below and have modified by adding i2c communication to an mcp23017, that all works have button registers and an led flashing on a timer.
I don't know how to implement 10bit on the analog axis or really how to modify the HID and USBAPI files included. the idea is to bump it all up to 7 10bit axis and 32 buttons.

Admin Post author
March 16, 2014 at 11:26 PM

I don't know what demo code you are talking about.

Did you define your struct yet? Do that first. Then determine the "usage page" and "usage" for the axis in a similar manner as the tutorial. Obviously set the report size to the correct number.

It's easier to up convert your data to be 16 bits to avoid padding bits.

It helps to write a portion of the descriptor, and then think "if I was a computer, how would I interpret this using the information available and the rules I need to follow".

Gavin
March 16, 2014 at 11:31 PM

apologies i was meant to link through.

http://forum.freetronics.com/viewtopic.php?f=27&t=734&start=10

Gavin
March 17, 2014 at 12:05 AM

Right this is what I have so far, taken a break and broke my chain of thought.

0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x04, // USAGE (Joystick)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x04, // REPORT_ID (4)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x01, // USAGE (Pointer)
0xa1, 0x00, // COLLECTION (Physical)
0x09, 0x30, // USAGE (X)
0x09, 0x31, // USAGE (Y)
0x09, 0x32, // USAGE (Z)
0x09, 0x36, // USAGE (Slider)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0x00, 0x04, // LOGICAL_MAXIMUM (1024)
0x75, 0x0a, // REPORT_SIZE (10)
0x95, 0x04, // REPORT_COUNT (4)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x20, // USAGE_MAXIMUM (Button 32)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x75, 0x01, // REPORT_SIZE (1)
0x95, 0x20, // REPORT_COUNT (32)
0x55, 0x00, // UNIT_EXPONENT (0)
0x65, 0x00, // UNIT (None)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0 // END_COLLECTION

does this part look right to you

Admin Post author
March 17, 2014 at 7:31 AM

Nope, you did not pad your 10 bit values, which makes it difficult to create a struct that represents your report. Either add the padding to the descriptor, or change the descriptor to say 16 bits instead and pad the value in the struct.

Gavin
March 16, 2014 at 7:04 PM

I'm using borrowed code from freetronics forum and running on a arduino leonardo.
The part i'm using is the third post from the bottom, i'm literally day two into testing parts of the code.
Might have to write all i have learnt up and make a nice joystick tutorial, no body seems to use higher the higher resolutions the ADC can provide.
Thanks for the reply, Gavin.

Gavin
March 17, 2014 at 9:34 AM

I think I have this right now

```
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x04, // USAGE (Joystick)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x04, // REPORT_ID (4)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x01, // USAGE (Pointer)
0xa1, 0x00, // COLLECTION (Physical)
0x09, 0x30, // USAGE (X)
0x09, 0x31, // USAGE (Y)
0x09, 0x32, // USAGE (Z)
0x09, 0x36, // USAGE (Slider)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0x00, 0x04, // LOGICAL_MAXIMUM (1024)
0x75, 0x0a, // REPORT_SIZE (10)
0x95, 0x04, // REPORT_COUNT (4)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x95, 0x01, // REPORT_COUNT (1)
0x75, 0x06, // REPORT_SIZE (6)
0x81, 0x03, // INPUT (Cnst,Var,Abs)
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x20, // USAGE_MAXIMUM (Button 32)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x75, 0x01, // REPORT_SIZE (1)
```

0x95, 0x20, // REPORT_COUNT (32)
0x55, 0x00, // UNIT_EXPONENT (0)
0x65, 0x00, // UNIT (None)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0 // END_COLLECTION

Admin Post author
March 17, 2014 at 2:52 PM

No, you've defined 10 bits 4 times, and then another 6 bits of padding, giving you 46 in total, which is not divisible by 8. You want 10, then 6, then 10 again then 6 again, then 10 again then 6 again, then 10 again then 6 again.

That's why it's easier to just define 16 bit 4 times instead.

gavin
March 17, 2014 at 3:31 PM

Ok I see, I have finally got my head around inserting all the pieces into the right places when adding an extra axis. The problem I have is no reference to what I need to add to make it a 16bit statement in hid descriptor and in the struct. Thanks for being patient, Gavin

locodog
March 18, 2014 at 12:51 AM

Also Gavin your logical maximum is 0x FF 03 (1023) not 1024,

Gavin
March 18, 2014 at 10:03 AM

Just stripping one axis out and modifying it should look like this then for a 16bit descriptor, would this also still be usable if i decided to use a 12bit or 16 bit adc?

0x05, 0x02, // USAGE_PAGE (Simulation Controls)
0x09, 0xbb, // USAGE (Throttle)

0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0xff, 0x03, // LOGICAL_MAXIMUM (1023)
0x75, 0x10, // REPORT_SIZE (16)
0x95, 0x01, // REPORT_COUNT (1)
0x81, 0x02 // INPUT (Data,Var,Abs)

and the stuct would use instead of “uint8_t throttle,” “uint16_t throttle,” and that will work?

Admin Post author
March 18, 2014 at 6:13 PM

At a glance, it should work.

Gavin
March 18, 2014 at 7:44 PM

No joy with that, pardon the pun. It will upload with no error but will not provide input. think it may be the struct.
This it just snippets of the code i have changed

```
//////////////////////////////////////////////////////////////////////
0x05, 0x02, // USAGE_PAGE (Simulation Controls)
0x09, 0xbb, // USAGE (Throttle)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0xff, 0x03, // LOGICAL_MAXIMUM (1023)
0x75, 0x10, // REPORT_SIZE (16) // putting this as 16 bit for
0x95, 0x01, // REPORT_COUNT (1) // future ADC upgrade
0x81, 0x02 // INPUT (Data,Var,Abs)
//////////////////////////////////////////////////////////////////////
Joystick_::Joystick_()
{
}

void Joystick_::move(uint8_t x, uint8_t y, uint8_t Rudder, uint16_t throttle, uint8_t buttons)
{
u8 j[5];
j[0] = x;
j[1] = y;
j[2] = Rudder;
```

```
j[3] = throttle;
j[4] = buttons;
//HID_SendReport(Report number, array of values in same order as HID descriptor, length)
HID_SendReport(4, j, 5);
}
//////////////////////////////////////////////////////////////////////
class Joystick_
{
public:
Joystick_();
void move(uint8_t x,
uint8_t y,
uint8_t Rudder,
uint16_t throttle,
uint8_t buttons);
};
extern Joystick_ Joystick;
//////////////////////////////////////////////////////////////////////
```

Admin Post author
March 18, 2014 at 11:23 PM

You are not really using that struct, you are trying to pack 16 bit variables into a 8 bit array. You need to learn to use type casting and cast your struct into an array, not use a separate array.

Gavin
March 18, 2014 at 11:52 PM

I am proper stuck

Gavin
March 19, 2014 at 9:52 AM

could you show me how to put that into practice?

Admin Post author
March 22, 2014 at 4:35 PM

research typecasting structures to byte arrays

NicoHood
March 21, 2014 at 6:45 PM

Hi, super tutorial, thanks!
signed int goes from −128 to 127, you can correct this if you want
http://en.wikipedia.org/wiki/Integer_%28computer_science%29#Common_integral_data_types

Admin Post author
March 22, 2014 at 4:34 PM

This is true but it's a preference of mine to keep it between -127 and +127 for the sake of keeping 0 right in the middle.

NicoHood
March 22, 2014 at 5:37 PM

Okay, thats a good point

I played a bit with the descriptors, but i dont really understand the whole thing. Its so complicated :/
I tried to add a System shutdown or sleep functionality, but it doesnt work. Any idea whats wrong here? i tried different version, nothing was working. or is just my win7 the problem?

```
char ReportDescriptor[26] = {
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x80, // USAGE (System Control)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x05, // REPORT_ID (5)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x19, 0x00, // USAGE_MINIMUM (Undefined)
0x29, 0x8d, // USAGE_MAXIMUM (System Menu Down)
0x95, 0x01, // REPORT_COUNT (1)
```

```
0x75, 0x10, // REPORT_SIZE (16)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0xff, 0x03, // LOGICAL_MAXIMUM (1023)
0x81, 0x00, // INPUT (Data,Ary,Abs)
0xc0 // END_COLLECTION
};
```

And i found another working code for Media keys like play/pause. Its working, i understood it but the line with 0x00, 0x00 i dont understand. it is nesessary, but why?

```
0x05,0x0C, // usage page (consumer device)
0x09,0x01, // usage — consumer control
0xA1,0x01, // collection (application)
0x85,0x04, // report id (4)

0x05,0x0C, // usage page (consumer)
0x19,0x00, // usage minimum (0)
0x2A,0xFF,0x03, //usage maximum (3ff)
0x95,0x01, //report count (1)
0x75,0x10, //report size (16)
0x15,0x00, //logical minimum
0x27,0xFF,0x03, //logical maximum (3ff)
0x00,0x00,
0x81,0x00, //input

0xC0,//end collection
```

thanks for you answer and thanks for the tutorial. its the only tutorial out there i could find. and these datasheets are so complicated for me, also with this technical english :S

Admin Post author
March 24, 2014 at 4:35 AM

the two 0x00 seems to be a part of the “logical maximum” item

you can check using http://eleccelerator.com/usb-descriptor-and-request-parser/

muhittin kaplan
April 3, 2014 at 1:58 PM

hi
I interested in USB KEYBOARD with STM32. can you send me usb keyboard sample project with stm32 ?
best regards

muhittin.kaplan@gmail.com

Admin Post author
April 7, 2014 at 1:14 AM

Search for ST USB Library, which probably contains an example of an USB keyboard implementation.

Andrey
June 25, 2014 at 5:55 PM

Hi Frank,

First of all, thank you very much for the tutorial, it helps a lot.

I'm developing a kind of controller for home automation, with a bunch of relays and buttons, and I deciided to use HID to be able to go without all these drivers. So, I (ab)used the FEATUREs to send data back and forth, just like it is done in one of the examples at obdev. The state of my device consists of 4 structures, 64 bytes each, and there are 'commands' sent to the device, either 'short' (8 bytes) or 'long' (16 bytes), all including that one byte for the report id. So I've made my report descriptor to show six different features. Here is it:

```
PROGMEM char usbHidReportDescriptor[78] = {
0x06, 0x00, 0xff, // USAGE_PAGE (Generic Desktop)
0x09, 0x01, // USAGE (Vendor Usage 1)
0xa1, 0x01, // COLLECTION (Application)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x26, 0xff, 0x00, // LOGICAL_MAXIMUM (255)
0x75, 0x08, // REPORT_SIZE (8)

0x85, uncreq_query_eeprom, // REPORT_ID (uncreq_query_eeprom)
0x95, 0x7f, // REPORT_COUNT (127)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x80, 0x01, // FEATURE (Data,Ary,Abs,Vol,Buf)
```

```
0x85, uncreq_command, // REPORT_ID (uncreq_command)
0x95, 0x07, // REPORT_COUNT (7)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x80, 0x01, // FEATURE (Data,Ary,Abs,Vol,Buf)

0x85, uncreq_long_command, // REPORT_ID (uncreq_long_command)
0x95, 0x0f, // REPORT_COUNT (15)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x80, 0x01, // FEATURE (Data,Ary,Abs,Vol,Buf)

0x85, uncreq_zero_query_counters,// REPORT_ID (uncreq_zero_query_counters)
0x95, 0x10, // REPORT_COUNT (16)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0xa0, 0x01, // FEATURE (Data,Ary,Abs,NPrf,Vol,Buf)

0x85, uncreq_query_fullstate, // REPORT_ID (uncreq_query_fullstate)
0x95, 0x3f, // REPORT_COUNT (63)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x00, 0x01, // FEATURE (Data,Ary,Abs,Buf)

0x85, uncreq_query_configuration,// REPORT_ID (uncreq_query_configuration)
0x95, 0x3f, // REPORT_COUNT (63)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x00, 0x01, // FEATURE (Data,Ary,Abs,Buf)

0x85, uncreq_query_extrastate, // REPORT_ID (uncreq_query_extrastate)
0x95, 0x3f, // REPORT_COUNT (63)
0x09, 0x00, // USAGE (Undefined)
0xb2, 0x00, 0x01, // FEATURE (Data,Ary,Abs,Buf)

0xc0 // END_COLLECTION
};
```

Everything works fine both under Windows and Linux (under Linux, I use libusb). However, my device also sends to the host a kind of ascii stream, and under Linux I can read it like this:

```
return usb_interrupt_read(dev_hdl, 0x81, buf, 8, 0);
```

Under windows, the same should (?) be achieved with

```
return hid_read(dev_hdl, (unsigned char*)buf, 8);
```

but it doesn't work because (as far as I understand) my descriptor doesn't describe any INPUTs. However, I several times tried to add something to my descriptor so that it tells the host there's input as well, and every time Windows just refused to see my device at all, perhaps because of malformed report descriptor. Under Linux, libusb simply ignores all the descriptor-related stuff and works as it is told to, but under Windows, well, libusb-win32 is a monster I never actually convinced to work as I want.

So, what is the right way, that is, what should I add to my descriptor so that the host understands there's also an input stream?

Thanks a lot!

Admin Post author
July 15, 2014 at 2:28 AM

It is very hard for me to help you like this, adding an input item should be simple, it could be a very minor mistake that's causing the problem. Did you check the length of your descriptor and what actual length is transmitted?

Stephan
August 12, 2014 at 2:33 PM

Thanks for the wonderful tutorial, it really helped me get the ball rolling. Also thanks for linking to the source pages it really helps to understand WHY things work they way they do. I have how ever been banging my head against a wall here for the last week or two. I am using a teensy 2.0 board on windows8. after reading your tutorial here I decided to modify their descriptor to include a second joystick. this is what I ended up with :

```
static const uint8_t PROGMEM joystick_hid_report_desc[] = {
0x05, 0x01, // Usage Page (Generic Desktop)
0x09, 0x04, // Usage (Joystick)
0xA1, 0x01, // Collection (Application)
0xA1, 0x00, // Collection (Phy.)
0x85, 0x04,//ID
/* button and axis definitions for 1stcontroller */
0xC0, // End Collection
0xC0, // End Collection

0x05, 0x01, // Usage Page (Generic Desktop)
0x09, 0x04, // Usage (Joystick)
```

0xA1, 0x01, // Collection (Application)
0xA1, 0x00, // Collection (phy)
0x85, 0x05,//ID
/* button and axis definitions for 2nd controller */
0xC0, // End Collection
0xC0 // End Collection

when I plug the device in windows creates two controllers, and the 2nd one work perfectly, but the first one gives me a blank pages, as if no attributes have been defined for it.
the descriptor is identical for both joysticks with the exception of the ID field.

my data buffer contains the id field as the first byte.
i've tried using different IDs i.e (1,2 2,1 3,4 4,3 4,5 5,4 1,5 5,1) the second controller always works perfectly and the first one is always just an empty controller. I've got no idea where I messed up.

Does anyone have any suggestions for me?

Thanks in advance

Admin Post author
August 17, 2014 at 5:57 AM

What hardware are you using?

The reason why I wrote the tutorial with two report IDs is that V-USB does not support multiple interfaces, only a single interface. V-USB is only supported on AVRs, so if you are not using AVRs, you can try another method.

Another way of achieving a two player controller is to create a device with two interfaces. Basically define more interface descriptors and use more endpoints. It's a bit more tricky but it avoids having a confusing HID descriptors.

Stephan
August 18, 2014 at 10:20 AM

I am using the atmega32u4, I've been doing some deeper digging since I asked that question. I am currently implementing a solution from the ground-up (rather than hacking apart the example I have) I think I have a good idea where I messed up. I will post my solution when (if) I figure it out.

Admin Post author
August 19, 2014 at 4:12 AM

You should be using LUFA then, you should definitely consider using multiple interfaces.

Wesko
September 17, 2014 at 3:39 PM

Thanks for being the only website that DOES actually expain HID. Still looking for a tool that converts a certain message I design as an interface blob to a descriptor. For instance, I send volume,channel as 2 bytes. What would be the descriptor?

Admin Post author
September 21, 2014 at 6:23 PM

For volume control, I wrote a small tutorial for Adafruit, check the source code for this project

https://learn.adafruit.com/trinket-usb-volume-knob

You need to read the library code first, not the sketch code.

Wesko
September 25, 2014 at 3:19 PM

Thanks, awesome. I am still looking into reliable tooling. The spec is not so clear, logical_minimum is signed for instance. And I cant find the report_count() max value. It is quite annoying all info is spread out over the internet

Alex
October 26, 2014 at 9:29 AM

Hi, This is a really handy article. I recently used this to implement a mouse trackball using .net micro and expanded the XY axis to 32 bit as i have high resolution ADC's on my board. I had some questions about but-

ton limits and what not for a mouse in windows. The device im building has support for 64 or more buttons. I was reading about the windows mouse driver and they say it supports 5 buttons. Is it possible to support more than 5 buttons? I was actually able to implement a hid descriptor that shows up as a mouse and has a report defined for the states of the 64 buttons im just unsure if those additional buttons beyond the first 5 will continue to raise pressed events in windows when they are pushed. Would it better to define the mouse and then define another interface for the keypad. Really interested in your thoughts on this.

Admin Post author
October 27, 2014 at 6:47 PM

Different interface is much better. The reason why I wrote about using only one interface is that V-USB doesn't support multiple interfaces.

I'd design your device as if it was a keyboard instead. I think you need N-key rollover which is hard to accomplish but I think there are some open source code on GitHub for that stuff already. (I forgot where, sorry, I found it before while researching MX Cherry keyboards)

GG
November 23, 2014 at 11:53 AM

This is an amazing work, I had been breaking my head in understanding the HID Specs. Your webpage has given most of the clarity which i have been looking for. Thank you!!

Wesko
November 25, 2014 at 10:14 AM

After weeks of exploring the descriptor parts look clear to me. I am using Vedor page 1, and I want to transfer back and forth 64 bytes using (e.g. report id 3). What puzzles me is how to write this in a java Android app. I got it running, it recognizes the Arduino board I use, but very unclear is whether I should use controlTransfer(), and if so, what all parameters are. So far this keeps failing:

```
final int requestType = UsbConstants.USB_DIR_OUT
| UsbConstants.USB_TYPE_VENDOR
| USB_SETUP_RECIPIENT_INTERFACE;
final int request = USBRQ_HID_SET_REPORT;
final int reportId = 3;
```

```
final int value = USB_OUTPUT_REPORT + reportId;
final int length = 64;
final int msec = 50;

final byte[] buffer = new byte[length];
System.arraycopy(data, 0, buffer, 0, data.length);

final int result = mUsbConnection.controlTransfer(requestType,
request, value, 0, buffer, length, msec);
```

MaxFloat
April 27, 2015 at 12:06 PM

In my Android app I managed to write data using controlTransfer, but I still cannot read data from device. Here are my sources:

```
public void onClick(View v)
{
write_txt.setText("");
if(myDev == null)
{
write_txt.append("Device not ready\n");
return;
}
write_txt.append("Device is ready\n");

boolean b = myManager.hasPermission(myDev);
if(b)
write_txt.append("Access granted\n");
else
write_txt.append("Access denied\n");

UsbDeviceConnection connection = myManager.openDevice(myDev);
if(connection == null)
{
write_txt.append("Connection not present\n");
return;
}
write_txt.append("Connection is ready\n");

//Requests on endpoint zero are not supported by this class(UsbRequest); use controlTransfer(int, int, int, int, byte[], int, int) for endpoint zero requests instead.
```

```
final int requestType = UsbConstants.USB_DIR_OUT | UsbConstants.USB_TYPE_VENDOR;
final int request = USBRQ_HID_SET_REPORT;
int value = 3;

String str = “momo”;
byte[] bytes = str.getBytes();
int TIMEOUT = 0;

int res = connection.controlTransfer(requestType, request, value, 0, bytes, bytes.length, TIMEOUT );// ( pt,
bytes, bytes.length, TIMEOUT); //do in another thread

if(res < 0)
write_txt.append("Write failed\n");
else
write_txt.append("Write of "+ res+"bytes succeeds!\n");

connection.close();
}
```

What is your progress now?

Shahzad
December 11, 2014 at 1:16 AM

What happens if we have more than one IN endpoints on HID device. How the report descriptor will be written to identify which IN endpoint is being accessed ?
I have configured one device where I have two IN endpoints (address 0x81 & 0x84) I have no clue how to write report descriptor to get data from both of them. My current report descriptor is reading one Byte from the endpoint which is defined first (If I define 0x81 endpoint descriptor first and then 0x84, then I’ll get data from 0x81 endpoint and if define 0x84 first then I start receiving data from 0x84 endpoint)

Admin Post author
December 15, 2014 at 7:26 AM

It has nothing to do with the report descriptor, only endpoint descriptor and interface descriptor matters.

If you are coding for the host, it’s easy to just say “read endpoint 0x81”

If you are coding for the device, it’s easy to just say “send this out endpoint 0x81”

If you want to make the second endpoint work with a standard HID driver, then sorry, standard HID drivers don't usually work that way.

Shahzad
December 15, 2014 at 9:59 AM

Thanks. I was looking for this answer whether Standard HID can support or I'll have to write some customized code to periodically read from both IN Endpoints (by the way both are Interrupt end points). In case standard HID can not support to read from two interrupt IN endpoints then I'll look for customized driver option

Amir
November 7, 2015 at 2:39 PM

Shahzad, I have same problem I want to send multiple endpoints (64 bytes for each of them)... what I have to do?

ashkan
December 30, 2014 at 12:11 PM

Vielen dank

THETHE
January 9, 2015 at 2:14 PM

Hi, I'm making an Game pad and HID LED controller working with a program.
Source of the controller for the program is opened but it is for STM32.
http://cgit.jvnv.net/arcin/tree/main.cpp

I'm working on arduino leonardo, which support HID.
I managed to make input buttons work based of your code, by modding HID.cpp on arduino IDE.
But when I just put this code under collection(physical), it even doesn't appears as proper device.

usage_page(Ordinal),
usage(1),

collection(Logical),
usage_page(LEDs),
usage(Generic Indicator),
report_size(1),
report_count(1),
output(data, var, abs)
end collection
(of course in hex)

any ideas?

Admin Post author
January 12, 2015 at 10:52 PM

I don't know why it doesn't work, but I would suggest not bothering with specific usage pages, and instead, just make a catchall generic report descriptor instead and have the microcontroller handle the data after delivery.

It's not worth the effort to craft a custom descriptor such that it works perfectly with all operating systems. For common descriptors, it's easy, for custom ones, it's hard.

Drewol
January 27, 2015 at 9:54 AM

Hi, I'm trying to do the same thing. Did you figure something out?

Matlo
January 21, 2015 at 8:38 AM

Hi Frank,

I think it would be a good idea to add something about global items (HID spec 1.11 page 35).
Without knowing what global items are it might be difficult to make sense from some HID descriptors.
There's a good example here:
http://blogs.msdn.com/b/usbcoreblog/archive/2014/02/14/reducing-the-size-of-hid-descriptors.aspx

chris
February 18, 2015 at 9:02 AM

Hi,

Thanks for the article that made my life easier each time I had to explain HID details.

Most of time I am using HID for remote controls and not for standard keyboard or multimedia devices and I have a question I cannot answer myself.
Some key functions as Play, Pause, FFWD, RWD… can be described as keyboard keys (keyboard page) or as multimedia keys (consumer page).
Linux can interpret both declarations for most of those keys.
Is there any benefit to declare them in consumer page or in keyboard page?

Thanks,
Chris

Admin Post author
March 17, 2015 at 3:28 PM

I've tested this and found Consumer-page is much better for multimedia controls, simply speaking, I found out that Keyboard-page 0x80 for volume-up doesn't work while Consumer-page 0xE9 for volume-up works.

Nick
March 17, 2015 at 7:17 AM

Hi,
Thanks for posting such a nice document on USB interface. I want to built a simple joystick with throttle control. I want to use ATMega32U2 with two analog Potentiometer for X-axis and Y-axis and a third analog Potentiometer for throttle control. What I am unable to find is the V-USB does not specify which micro-controller to use. ATMega32U2 have a built-in USB interface. Kindly help me in getting my joystick developed on ATMega32U2.
regards,
Nick.

Admin Post author
March 17, 2015 at 3:23 PM

V-USB is for AVR microcontrollers WITHOUT built-in USB.

ATmega32U2 has built-in USB, so do NOT use V-USB with it. Use LUFA instead, which comes with example projects similar to your goal.

Nick
March 18, 2015 at 10:14 AM

Hi,
I have designed a circuit for usb joystick using ATMEGA8A-PU micro-controller similar to the one in the circuit directory of-V-USB latest source. I am having some issues using WinAVR with AVR Studio V.4.14 compiler to create a hex file that I can burn in the micro-controller. Currently I am using the hid-mouse example from V-USB. Can you please help me in getting the code compiled using winav?

regards,
Nick

Ash
March 25, 2015 at 1:49 AM

Hello Guys,

I am trying to implement Key press volume up feature for iOS using HID Consumer control, But its not working at all. Could you please share the descriptor (report map) for Volume up control.

Thanks n Regards,

/Ash

Admin Post author
March 25, 2015 at 2:01 AM

How are you even connected to iOS? I don't recall any iPhone or iPad having USB host capabilities at all.

If you need the data format, please infer it using this example https://github.com/adafruit/Adafruit-Trinket-USB/tree/master/TrinketHidCombo/ which I wrote while working for Adafruit

jeffery
April 20, 2015 at 2:12 PM

Hi Frank, I am trying to inject input into a virtual mouse device I created with the Microsoft virtual hid mini-port sample from Microsoft. The WDK forum guys recommended I pass the data that would be injected into the mouse devices from a TLC (top level collection) and add a custom usage page. My question is how would I perform this injection? I do not want to use writefile if possible but the Windows.Devices.HumanInterfaceDevice class methods/functions. The problem is that the injection has to be a custom device or a joystick similar to a mouse device so the data can be properly passed down. Here's my report descriptor so far:

```
HID_REPORT_DESCRIPTOR G_DefaultReportDescriptor[] = {
0x06, 0x00, 0xFF, // USAGE_PAGE (Vender Defined Usage Page)
0x09, 0x01, // USAGE (Vendor Usage 0x01)
0xA1, 0x01, // COLLECTION (Application)
0x85, CONTROL_FEATURE_REPORT_ID, // REPORT_ID (1)
0x09, 0x01, // USAGE (Vendor Usage 0x01)
0x15, 0x00, // LOGICAL_MINIMUM(0)
0x26, 0xff, 0x00, // LOGICAL_MAXIMUM(255)
0x75, 0x08, // REPORT_SIZE (0x08)
//0x95,FEATURE_REPORT_SIZE_CB, // REPORT_COUNT
0x96, (FEATURE_REPORT_SIZE_CB & 0xff), (FEATURE_REPORT_SIZE_CB >> 8), // REPORT_COUNT
0xB1, 0x00, // FEATURE (Data,Ary,Abs)
0x09, 0x01, // USAGE (Vendor Usage 0x01)
0x75, 0x08, // REPORT_SIZE (0x08)
//0x95,INPUT_REPORT_SIZE_CB, // REPORT_COUNT
0x96, (INPUT_REPORT_SIZE_CB & 0xff), (INPUT_REPORT_SIZE_CB >> 8), // REPORT_COUNT
0x81, 0x00, // INPUT (Data,Ary,Abs)
0x09, 0x01, // USAGE (Vendor Usage 0x01)
0x75, 0x08, // REPORT_SIZE (0x08)
//0x95,OUTPUT_REPORT_SIZE_CB, // REPORT_COUNT
0x96, (OUTPUT_REPORT_SIZE_CB & 0xff), (OUTPUT_REPORT_SIZE_CB >> 8), // REPORT_COUNT
0x91, 0x00, // OUTPUT (Data,Ary,Abs)
0xC0, // END_COLLECTION
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x02, // USAGE (Mouse)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x02, // REPORT_ID (2)
0x09, 0x01, // USAGE (Pointer)
0xa1, 0x00, // COLLECTION (Physical)
0x05, 0x09, // USAGE_PAGE (Button)
```

```
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x03, // USAGE_MAXIMUM (Button 3)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x95, 0x03, // REPORT_COUNT (3)
0x75, 0x01, // REPORT_SIZE (1)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x95, 0x01, // REPORT_COUNT (1)
0x75, 0x05, // REPORT_SIZE (5)
0x81, 0x03, // INPUT (Cnst,Var,Abs)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x30, // USAGE (X)
0x09, 0x31, // USAGE (Y)
0x15, 0x81, // LOGICAL_MINIMUM (-127)
0x25, 0x7f, // LOGICAL_MAXIMUM (127)
0x75, 0x08, // REPORT_SIZE (8)
0x95, 0x02, // REPORT_COUNT (2)
0x81, 0x06, // INPUT (Data,Var,Rel)
0xc0, // END_COLLECTION
0xc0, // END_COLLECTION
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x02, // USAGE (Mouse)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x03, // REPORT_ID (2)
0x09, 0x01, // USAGE (Pointer)
0xa1, 0x00, // COLLECTION (Physical)
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x03, // USAGE_MAXIMUM (Button 3)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x95, 0x03, // REPORT_COUNT (3)
0x75, 0x01, // REPORT_SIZE (1)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x95, 0x01, // REPORT_COUNT (1)
0x75, 0x05, // REPORT_SIZE (5)
0x81, 0x03, // INPUT (Cnst,Var,Abs)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x30, // USAGE (X)
0x09, 0x31, // USAGE (Y)
0x15, 0x81, // LOGICAL_MINIMUM (-127)
0x25, 0x7f, // LOGICAL_MAXIMUM (127)
0x75, 0x08, // REPORT_SIZE (8)
0x95, 0x02, // REPORT_COUNT (2)
```

```
0x81, 0x06, // INPUT (Data,Var,Rel)
0xc0, // END_COLLECTION
0xc0, // END_COLLECTION
};
```

Andrew
May 28, 2015 at 2:58 AM

Thanks for a wonderful description. For a year, I have been waiting for Adafruit to make a V-USB library for turning their Trinket (ATtiny85) into a gamepad. With some free time this summer, I have finally decided to try making the libraries I have sought myself. However, even though I have the source codes for a couple different related implementations, I could not figure out the HidReportDescriptors until I found your post. Now they make sense (or at least enough to move forward)!
Thanks from a college EE-in-Training and hobbyist.

-Andrew

Andrew
May 28, 2015 at 3:02 AM

Right after posting, I just realized that you wrote the Trinket V-USB libraries I am working from. (Also explains why I saw the USB business card you have on your this blog on Adafruit's blog a few years ago)
So I guess I should rephrase it as "with your blog and working from your code (and others) I hope to create a new library and implementation."

iordtsin
June 22, 2015 at 1:28 PM

Hi,
Thanks for the great and wonderfull tutorial.
I want to make an application which uses a 3 button mouse the sent data should be like this Button(1Byte) X(2Bytes) Y (2Bytes) Wheel(1 Byte).
a help would be nice cause i tried out to find it myself for 2 weeks and i have still problem .
Thanks in advance!!!

Teiwaz
June 27, 2015 at 8:36 PM

I'm working on a similar project with a Leonardo, but am having trouble with the HID descriptor defining multiple reports on multiple application collections.

Here is my hacked-down descriptor:

```
const u8 _hidReportDescriptor[] = {
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x04, // USAGE (Joystick)
0xa1, 0x01, // COLLECTION (Application)
0x85, 0x01, // REPORT_ID (1)
0xa1, 0x00, // COLLECTION (Physical)
//Buttons:
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
0x29, 0x20, // USAGE_MAXIMUM (Button 32)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x75, 0x01, // REPORT_SIZE (1)
0x95, 0x20, // REPORT_COUNT (32)
0x55, 0x00, // UNIT_EXPONENT (0)
0x65, 0x00, // UNIT (None)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0, // END_COLLECTION (Physical)

/*
0xc0, // END_COLLECTION (Application)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x04, // USAGE (Joystick)
0xa1, 0x01, // COLLECTION (Application)
*/
0x85, 0x02, // REPORT_ID (2)
0xa1, 0x00, // COLLECTION (Physical)
//Buttons:
0x05, 0x09, // USAGE_PAGE (Button)
0x19, 0x01, // USAGE_MINIMUM (Button 1)
```

```
0x29, 0x20, // USAGE_MAXIMUM (Button 32)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x01, // LOGICAL_MAXIMUM (1)
0x75, 0x01, // REPORT_SIZE (1)
0x95, 0x20, // REPORT_COUNT (32)
0x55, 0x00, // UNIT_EXPONENT (0)
0x65, 0x00, // UNIT (None)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0, // END_COLLECTION (Physical)
0xc0 // END_COLLECTION (Application)
};
```

This version is accepted, but windows sees it as a single joystick with 64 buttons (only 32 of which can be read due to limitations in windows.) If you remove the comment block in the middle (splitting the two reports into two separate application collections) the device fails to start, error 10.

I've gone through your tutorial a few times and haven't seen anything that seems like it's special setup to allow windows to accept multiple application collections. Are you aware of anything special that needs to be done to allow this?

Ali
June 30, 2015 at 4:21 PM

Hello.
i have a joystick logitech 3d pro . can you plz help me with that ??
thank you.

Tanakhon
October 6, 2015 at 5:49 PM

Sorry i am a newbie, How to run the project?

zuby
November 14, 2015 at 4:30 PM

Hi,
I wrote HID descriptor for Joystick, the joystick has 8 axis. I want to know, is this right that Wheel and Dial usage for trim wheel and flap axis and how can I add 8 buttons into the below code I have many tried for buttons but failed?
sorry for my bad english.

```
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x09, 0x04, // USAGE (Joystick)
0xa1, 0x01, // COLLECTION (Application)
0x05, 0x02, // USAGE_PAGE (Simulation Controls)
0x09, 0xbb, // USAGE (Throttle)
0x15, 0x81, // LOGICAL_MINIMUM (-127)
0x25, 0x7f, // LOGICAL_MAXIMUM (127)
0x75, 0x08, // REPORT_SIZE (8)
0x95, 0x01, // REPORT_COUNT (1)
0x81, 0x02, // INPUT (Data,Var,Abs)
0x05, 0x01, // USAGE_PAGE (Generic Desktop)
0x09, 0x01, // USAGE (Pointer)
0xa1, 0x00, // COLLECTION (Physical)
0x09, 0x30, // USAGE (X) for ailerons
0x09, 0x31, // USAGE (Y) for elevator
0x09, 0x32, // USAGE (Z) for rudder
0x09, 0x33, // USAGE (Rx) for spoiler
0x09, 0x34, // USAGE (Ry) for throttle 1
0x09, 0x35, // USAGE (Rz) for throttle 2
0x09, 0x38, // USAGE (Wheel) for trim wheel
0x09, 0x37, // USAGE (Dial) for flap
0x95, 0x08, // REPORT_COUNT (8)
0x81, 0x02, // INPUT (Data,Var,Abs)
0xc0, // END_COLLECTION
0xc0 // END_COLLECTION
```

suresh
January 27, 2016 at 6:47 AM

Nice article!!!

Suresh Gupta
February 4, 2016 at 3:09 AM

Hi Great article.
I have a different requirement and that is I want to transfer some file of 1K bytes size from host to device on USB HID. Do you think it is possible and if so what will be the descriptor in that case?
I want to use HIDAPI at host side to provide 1K bytes in one go. Do you think the HIDAPI will break this data in 64 bytes chunks and how I capture this at gadget side if I am reading from /dev/hidg*

Thanks

Soren
February 20, 2016 at 9:55 PM

Hi,

Great introduction to HID Descriptors, it has helped me a few times.

I realize the maximum report size is limited to 64 bytes, but is that per HID descriptor, or can I specify multiple input reports in one HID descriptor, where the combined size exceed 64 bytes?

Each input report is transmitted separately.

1rel
March 25, 2016 at 11:03 PM

Thanks, that was really helpful to get my STM32F042 based controller working...

Henrik
January 5, 2017 at 8:11 PM

Can you share som insight on your F042 USB problems? Having Device descriptor fail errors with CubeMX generated code (CDC VCP class).

Matheus
March 30, 2016 at 1:05 AM

Hello,
I'm developing an ArcadePad with a PIC18F4550 to be recognized as a game controller (joystick). I've changed the descriptor in order to have 12 buttons, the hat switch and the X,Y axis. All the simulation worked good, the PIC was detected as a game controller but I always get all the bytes with 0x00, whatever the value I send to the PC. Does anybody know what could be happening? I'm using MikroC USB libraries.

Uzair Ali
April 30, 2016 at 10:01 PM

Nicely explained, I want to know how to implement Force Feedback feature all the information is available in physical interference device in http://www.usb.org/developers/hidpage#Physical_Interface website but I couldn't able to follow that. It would be great if you help me.
Thanks

Ari Zagnoev
June 24, 2016 at 5:49 AM

Thank you for this tutorial.
I am interested in understanding the keyboard descriptor but it seems to have been cut out. You refer to it but it isn't there
I am working on the Combimouse http://www.combimouse. com which is a split keyboard. My problem is that the command keys (ie. Shift, Ctrl, Alt, GUI) are only sent when another key is pressed. With my split keyboard this is a problem. For example if I press the Shift key on the left keyboard and a key on the right keyboard it isn't capitalised.

I see that there are the USB codes E0 to E7 for the command keys and I am trying to use them but it's not working. My first step is to make sure my descriptor is okay.

Amanpreet Singh
July 21, 2016 at 6:06 AM

Hey,
It is a nice article and very helpful.
I think your Mouse Data Report Byte0 for buttons is not correct. Bit0 should be for Left button, Bit1 for Right button and Bit2 for Middle button. I have tested it in USB2.0 Mouse.
Please correct me if I am wrong.
Thanks for such a helpful tutorial.

Great work
Amanpreet

José Pablo Téllez Vázquez
December 27, 2016 at 7:06 PM

Hi Friend, I only want to thank you for this post, because it has been very useful for me to modify usb descriptor for gamepad on STM32F103C8T6 microcontroller, note that after following this tutorial, just left to changing the PID adding 1. thank you very much!

Naushad rahman
January 17, 2017 at 7:08 AM

awesome tutorial ....was really helpful

John Madsen
February 2, 2017 at 4:00 AM

AWESOME tutorial! Figuring this out was making me crazy! Thank you SO much!