Results 1 to 20 of 20

Thread: Using Serial and MIDI USB types at the same time

07-12-2013, 01:55 AM #1

Dark_eye

Junior Member

Join Date: Jul 2013

Posts: 6

Using Serial and MIDI USB types at the same time

Hi all!

Before I say a single word I want to congratulate Paul and all the Teensy community for the hard work done, I am studying Telco Ing. and I always been interested in electronics and networks. Finding something like Teensy clearly made me order one of your boards inmediatly, although having looked at Arduino, which is both worse and more expensive.

Going thru my case, I have my Teensy shipped but It didnt arrived yet. So I am only asking based on what I've learned from the teensyduino section of this page, and some Arduino code from the Internet. And I also found another similar post in this forums, but didnt have any reply...

My question is, Can I use the USB serial communications while I am using USB MIDI? This is for a MIDI controller for Guitar Rig, a virtual-amp VST plugin.
I have a program that reads in which preset is active and sends 4 ASCII chars back to my Teensy for displaying preset name on the controlloler's 4-digit 7-seg display.

I have seen in code samples from this page that USB MIDI is programmed only by calling usbMIDI.sendNoteOn() and other functions directly from the loop function. What may happen if you also call Serial.begin(9600); on setup and use the serial communications functions?
Maybe the limits arise when you have to select USB type at Arduindo IDE... I've heard about composite USB types, and I am kind of programming something, as I've seen before here.

If this is completely imposible, I can design a "protocol" for sending back the ASCII data encoded as MIDI notes

Greetings !

Reply With Quote

07-12-2013, 08:31 PM #2

PaulStoffregen

Senior Member

Join Date: Nov 2012
Posts: 23,258

It is possible to have both USB serial and MIDI, but Teensyduino currently does not have any option to implement this. You'll need to modify the USB code, which involves digging into the core library and hacking the code. It's much easier on Teensy 3.0 (see the comments in usb_desc.h), but also possible on 2.0 with some programming.

Without editing the core library, you still have a few options.

Teensyduino emulates a serial connection using HID for all the USB types without Serial. So you can put Serial.print() in your program and it will print to the Arduino Serial Monitor. This might be useful, but it's not simple to access without the serial monitor, because it not a serial device (Mac, Linux) or COM port (Windows). There are 2 ways to access it. #1: You can use the "teensy_gateway" program and then telnet to 127.0.0.1 port 28541. That's how the serial monitor does it when using a USB type without serial. #2: You could use the native HID APIs to directly receive the data. Each operating system has very different APIs. The hid_listen program is probably a good starting point.

Within MIDI, the System Exclusive message is the normal way for sending these types of non-MIDI messages. If your software can deal with sysex messages, that might be the easiest path. It's probably much easier than using note on/off or other normal MIDI messages to send ASCII data.

Another option, which probably isn't very desirable for USB, would be using the hardware serial port. It's completely independent of USB. To get it back into your computer, you'd need a separate USB-serial converter. But you mentioned a goal of "displaying preset name on the controlloler's 4-digit 7-seg display". If that's an on-screen display on the computer, then of course you need to get the data back in... but if it's another piece of hardware, perhaps just sending the data directly to the hardware with that display might be simpler than sending it to the computer which then retransmits it to the display.

Of course, if it's a dedicated display, you might also just be able to wire the display to some I/O pins on the Teensy?

Reply With Quote

07-13-2013, 02:45 AM #3

Dark_eye

Junior Member

Join Date: Jul 2013
Posts: 6

Thanks for you response Paul,

Im actually using Teensy 3.0 (Not yet, still waiting...)

I thing you misunderstood my display configuration, the 4-

digit 7-seg display is connected to Teensy along with the switches, and I am searching a way to send preset name from Guitar Rig (pc) to Teensy, and have the name displayed (something poor, because being a 7-seg) on the display.

So, as I like programming a lot, I think I'll go the path of editing the USB code. Thanks for the clue of the C file, I'll take a look at that.
I'll post results and instructions once I get it to work, and my Teensy arrives!

Greets!

Reply With Quote

07-20-2013, 02:38 AM #4

Dark_eye

Junior Member
Join Date: Jul 2013
Posts: 6

Hi again Paul,

This has become more challenging than initially thought.

I have modified the USB_MIDI type in order to use the normal Serial mode along with the MIDI, instead of the seremu serial. (It would be great if you can explain what is exactly the serial emulator)

The USB_desc.h is as follows:

Code:

```
#elif defined(USB_MIDI)
  #define VENDOR_ID             0x16C0
  #define PRODUCT_ID            0x0485
  #define MANUFACTURER_NAME     {'T','e','e','
  #define MANUFACTURER_NAME_LEN 11
  #define PRODUCT_NAME          {'T','e','e','
  #define PRODUCT_NAME_LEN      11
  #define EP0_SIZE              64
  #define NUM_ENDPOINTS         8
  #define NUM_USB_BUFFERS       28
  #define NUM_INTERFACE         3

  #define CDC_STATUS_INTERFACE  0
  #define CDC_DATA_INTERFACE    1
  #define CDC_ACM_ENDPOINT              2
  #define CDC_RX_ENDPOINT       3
  #define CDC_TX_ENDPOINT       4
  #define CDC_ACM_SIZE          16
  #define CDC_RX_SIZE           64
  #define CDC_TX_SIZE           64

  #define MIDI_INTERFACE        2       // MIDI
  #define MIDI_TX_ENDPOINT      5
  #define MIDI_TX_SIZE          64
  #define MIDI_RX_ENDPOINT      6
  #define MIDI_RX_SIZE          64

  #define CONFIG_DESC_SIZE      (9 +9+5+5+4+5+7

  #define ENDPOINT2_CONFIG      ENDPOINT_TRANSI
```

I struggled to find a sense for every bit of configuration, so probably I done something wrong; mainly because with this configuration Windows only detects one MIDI interface, and another Unidentified HID device, and with errors.

Obviously, I also had to modify files usb_inst.cpp:

Code:

```
#ifdef USB_MIDI
usb_midi_class usbMIDI;
usb_serial_class Serial;
#endif
```

In order to use normal USB serial and not Serial Emulator

usb_serial.h:

Code:

```
#if defined(USB_SERIAL) || defined(USB_SERIAL_H

#include <inttypes.h>

// C language implementation
#ifdef __cplusplus
extern "C" {
#endif
int usb_serial_getchar(void);
int usb_serial_peekchar(void);
int usb_serial_available(void);
int usb_serial_read(void *buffer, uint32_t size
void usb_serial_flush_input(void);
```

In order to have the normal serial code compiled when the MIDI interface is used

And usb_seremu.h:

Code:

```
#ifndef USBseremu_h_
#define USBseremu_h_

#if defined(USB_HID) || defined(USB_RAWHID) ||

#include <inttypes.h>

// C language implementation
#ifdef __cplusplus
extern "C" {
#endif
int usb_seremu_getchar(void);
int usb_seremu_peekchar(void);
int usb_seremu_available(void);
```

For not using it when MIDI is selected

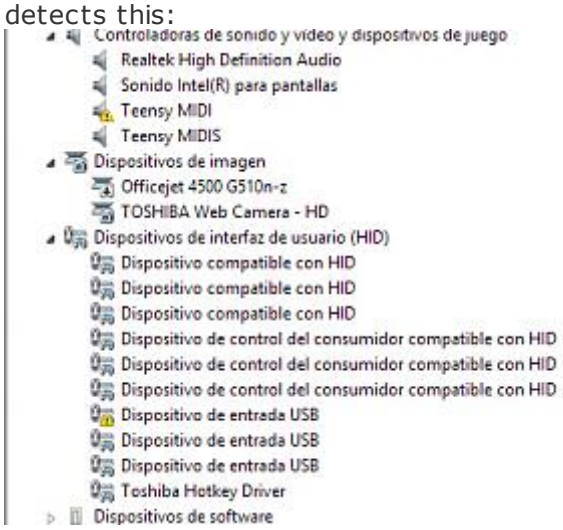
So, I was running the following sketch, in order to test Serial and MIDI together:

Code:

```
const int channel = 1;

void setup()
{
  Serial.begin(9600);
  pinMode(led, OUTPUT);

}

void loop()
{
  tocar(60,500);
  tocar(62,500);
  tocar(64,500);
  tocar(66,500);
  tocar(68,500);
  tocar(70,5000);
}


void tocar (int nota, int duracion)
```

```
void tocar (int nota,int duracion)
{
  usbMIDI.sendNoteOn(nota, 99, channel);  // 6(
  digitalWrite(led,HIGH);
  Serial.println("Tocando nota");
  delay(duracion);
  usbMIDI.sendNoteOff(nota, 0, channel);  // 6(
  digitalWrite(led,LOW);
  delay(200);

}
```

Everything compiles ok, runs ok, but Windows just detects this:



EDIT: The double Teensy MIDI Interface is because I initially named it differently, only for the sake of trying; reverted to original name when I realised it didn't work, and ¿maybe Windows cached it?

So, that's where I reached, with my zero USB programming knowledge I will need some help from an expert.

Greetings and thanks in advance.

Last edited by Dark_eye; 07-20-2013 at 02:42 AM. **Reason:** Forgot a little detail

Reply With Quote

07-22-2013, 06:09 AM #5

PaulStoffregen

Senior Member

Join Date: Nov 2012
Posts: 23,258

Yes, windows caches lots of stuff in its registry, even though it needs to read all that data anyway to detect the device.

Every time you make a change, increase the bcd device rev number. That will let windows know its new and to not use the cached info.

Reply With Quote

07-26-2013, 12:52 AM #6

Dark_eye

Junior Member

Join Date: Jul 2013

Posts: 6

Hi again Paul,

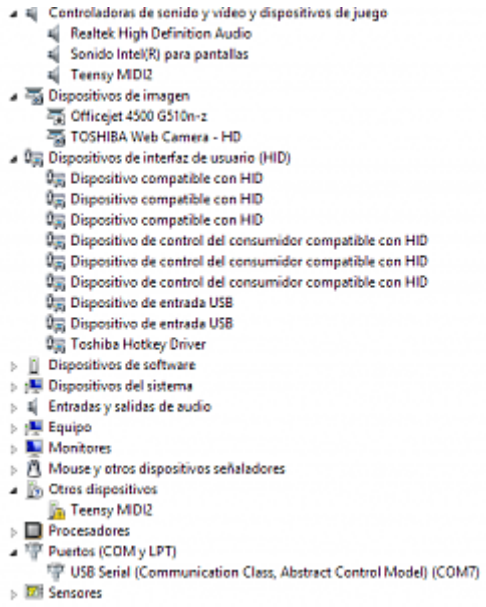
Unfortunately, after playing and messing with it, I can't manage to do it.

My usb_desc.h is as follows:

Code:

```
#elif defined(USB_MIDI)
  #define VENDOR_ID              0x16C0
  #define PRODUCT_ID             0x0485
  #define MANUFACTURER_NAME      {'T','e','e','n
  #define MANUFACTURER_NAME_LEN 11
  #define PRODUCT_NAME           {'T','e','e','n
  #define PRODUCT_NAME_LEN       12
  #define EP0_SIZE               64
  #define NUM_ENDPOINTS          6
  #define NUM_USB_BUFFERS        30
  #define NUM_INTERFACE          3
  #define CDC_STATUS_INTERFACE   0
  #define CDC_DATA_INTERFACE     1  // Serial
  #define CDC_ACM_ENDPOINT              2
  #define CDC_RX_ENDPOINT        3
  #define CDC_TX_ENDPOINT        4
  #define CDC_ACM_SIZE           16
  #define CDC_RX_SIZE            64
  #define CDC_TX_SIZE            64

  #define MIDI_INTERFACE         2       // MIDI
  #define MIDI_TX_ENDPOINT       5
  #define MIDI_TX_SIZE           64
  #define MIDI_RX_ENDPOINT       6
  #define MIDI_RX_SIZE           64

  #define CONFIG_DESC_SIZE       (9 +9+5+5+4+5+7

  #define ENDPOINT2_CONFIG       ENDPOINT_TRANSI
  #define ENDPOINT3_CONFIG       ENDPOINT_RECEIV
```

Uninstalling all the previous USB devices from Windows, and altering bcdDevice leads to this results, which are close:



-The MIDI system is working fine
-The Serial system is NOT working at all

If I open the COM7 with putty I cant see anything coming from teensy, and when tried to send to teensy, an error shows saying: "Error writting to serial device".

If you could provide some directions to fix this I would be

very pleased.

Greetings

Reply With Quote

04-15-2015, 04:38 PM #7

adrianfreed

Senior Member

Join Date: Mar 2013
Posts: 131

I was just wondering if you got this working yet?

Reply With Quote

04-15-2015, 05:08 PM #8

Dark_eye

Junior Member

Join Date: Jul 2013
Posts: 6

> Originally Posted by **adrianfreed**
> I was just wondering if you got this working yet?

Hi! So much time after whew

I finally used one of the Paul's ideas. I wrote the management program to connect via TCP Socket (127.0.0.1 port 28541) to the server opened by the "teensy_gateway" binary program. So the Serial-Style communication happens over TCP, just the same logic and protocol, but using Sockets instead of serial.

The only hassle of that approach is to check if the port is open and if not, to execute the "teensy_gateway" executable beforehand. Though my program managed all of that automatically

If you have some more questions, just ask.

Greetings!

Reply With Quote

04-15-2015, 06:53 PM #9

PaulStoffregen

Senior Member

Join Date: Nov 2012
Posts: 23,258

Since this thread started, Teensyduino 1.21 and 1.22 have been released.

1.21 made changes to usb_desc.h to simplify the process of creating custom USB configurations. No longer do you need to compute those error-prone offsets and lengths. Most of the other code now automatically adapts to usb_desc.h too, so it's much simpler to just edit what you want in one location.

The instructions in the comments were also improved.

Reply With Quote

04-15-2015, 07:06 PM #10

mlu

Senior Member

In this situation I would have used some MIDI sysex message or some available controller MIDI message to the

Join Date: Aug 2013
Location: Gothenburg, Sweden
Posts: 337

Teensy, letting the the Teensy convert this to display data.

Reply With Quote

05-29-2015, 02:29 PM #11

monkeybiscuits

Member
Join Date: Dec 2014
Location: Japan
Posts: 37

> Originally Posted by **Dark_eye**
> If you have some more questions, just ask.
> Greetings!

Would your program allow Teensy to **send** keystrokes and MIDI messages simultaneously? I have a MIDI controller that I'd love an 'undo' or 'delete' button for. I know there's a great program for mac (midiStroke) but I'm on a windows machine. If it is possible, would you be willing to share your code?

UPDATE: SUCCESS!!! I just had a little trouble finding the 'usb_desc.h' file (Arduino\hardware\teensy\avr\cores\teensy3) but, once I found it, I followed Paul's excellent comments step-by-step and it worked like a charm the first time. In my case I just copied lines 157~160 and pasted them down into the USB_MIDI section, changed all the necessary numbers according to Paul's instructions and added the following line to the end of that section:

Code:

```
#define ENDPOINT5_CONFIG  ENDPOINT_TRANSIMIT_
```

(because 5 was the number I used for the new endpoint and I only need it to transmit.

In the sketch, I needed a new include

Code:

```
#include <usb_keyboard.h>
```

and that's it! You can call any keystroke combination you want with

Code:

```
usb_keyboard_press(KEY_DELETE, 0);
```

for example.

Last edited by monkeybiscuits; 11-23-2015 at 10:01 AM. **Reason:** UPDATE: SUCCESS!!!

Reply With Quote

12-11-2015, 06:49 AM #12

sjtrny

Junior Member

Join Date: Dec 2015

Posts: 5

So I'm trying to do what the OP intended for my own project. I have edited the USB_MIDI def in usb_desc.h and MIDI works but I can't get any Serial.println()'s to show in the Arduino IDE serial monitor. Am I missing something?

EDIT: When I delete all the MIDI interface and endpoint references from the def THEN I can see the serial device listed in /dev and I have a serial terminal app for OS X but it is unable to open the device (it suggests it is in use already).

Here's my new def:

Code:

```
#elif defined(USB_MIDI)
    #define VENDOR_ID           0x16C0
    #define PRODUCT_ID          0x0485
    #define MANUFACTURER_NAME   {'T','e','e','r
    #define MANUFACTURER_NAME_LEN       11
    #define PRODUCT_NAME                {'T','e
    #define PRODUCT_NAME_LEN        11
    #define EP0_SIZE                64
    #define NUM_ENDPOINTS           5
    #define NUM_USB_BUFFERS         20
    #define NUM_INTERFACE           3
    #define MIDI_INTERFACE          0   // MIDI
    #define MIDI_TX_ENDPOINT        1
    #define MIDI_TX_SIZE            64
    #define MIDI_RX_ENDPOINT        2
    #define MIDI_RX_SIZE            64
    #define CDC_STATUS_INTERFACE        1   //
    #define CDC_DATA_INTERFACE      2
    #define CDC_ACM_ENDPOINT        3
    #define CDC_RX_ENDPOINT         4
    #define CDC_TX_ENDPOINT         5
    #define CDC_ACM_SIZE            16
    #define CDC_RX_SIZE             64
    #define CDC_TX_SIZE             64
    #define ENDPOINT1_CONFIG    ENDPOINT_TRANSI
    #define ENDPOINT2_CONFIG    ENDPOINT_RECEIV
    #define ENDPOINT3_CONFIG    ENDPOINT_TRANSI
    #define ENDPOINT4_CONFIG    ENDPOINT_RECEIV
    #define ENDPOINT5_CONFIG    ENDPOINT_TRANSI
```

Test code that I'm using (because I know some of you will think it's the problem):

Code:

```
void setup() {

  Serial.begin(115200);

}

void loop() {

  Serial.println("Hello");
}
```

Last edited by sjtrny; 12-11-2015 at 07:15 AM.

Reply With Quote

12-11-2015, 07:34 AM #13

monkeybiscuits

Member
Join Date: Dec 2014
Location: Japan
Posts: 37

I can't see a problem with yours but here's the one I use.

Code:

```
#elif defined(USB_MIDI)
  #define VENDOR_ID             0x16C0
  #define PRODUCT_ID            0x0485
  #define MANUFACTURER_NAME     {'T','e','e','r
  #define MANUFACTURER_NAME_LEN 11
  #define PRODUCT_NAME          {'T','e','e','r
  #define PRODUCT_NAME_LEN      11
  #define EP0_SIZE              64
  #define NUM_ENDPOINTS         5
  #define NUM_USB_BUFFERS       16
  #define NUM_INTERFACE         3
  #define SEREMU_INTERFACE      1        // Ser
  #define SEREMU_TX_ENDPOINT    1
  #define SEREMU_TX_SIZE        64
  #define SEREMU_TX_INTERVAL    1
  #define SEREMU_RX_ENDPOINT    2
  #define SEREMU_RX_SIZE        32
  #define SEREMU_RX_INTERVAL    2
  #define MIDI_INTERFACE        0        // MID
  #define MIDI_TX_ENDPOINT      3
  #define MIDI_TX_SIZE          64
  #define MIDI_RX_ENDPOINT      4
  #define MIDI_RX_SIZE          64
  #define KEYBOARD_INTERFACE    2        // Key
  #define KEYBOARD_ENDPOINT     5
  #define KEYBOARD_SIZE         8
  #define KEYBOARD_INTERVAL     1
  #define ENDPOINT1_CONFIG      ENDPOINT_TRANS
  #define ENDPOINT2_CONFIG      ENDPOINT_RECEI
  #define ENDPOINT3_CONFIG      ENDPOINT_TRANS
```

Reply With Quote

12-11-2015, 07:36 AM #14

sjtrny

Junior Member
Join Date: Dec 2015
Posts: 5

That's emulated serial though. I want to be able to access it as a real serial device.

Last edited by sjtrny; 12-11-2015 at 07:50 AM.

Reply With Quote

12-11-2015, 08:26 AM #15

sjtrny

Junior Member
Join Date: Dec 2015
Posts: 5

> Originally Posted by **sjtrny**
> That's emulated serial though. I want to be able to access it as a real serial device.

Ok I got it working. Copied the defs from USB_SERIAL_HID instead of USB_SERIAL and it works!

Code:

```
#elif defined(USB_MIDI)
#define VENDOR_ID             0x16C0
#define PRODUCT_ID            0x0485
#define MANUFACTURER_NAME     {'T','e','e','r
#define MANUFACTURER_NAME_LEN 11
#define PRODUCT_NAME          {'T','e','e','r
#define PRODUCT_NAME_LEN      11
#define EP0_SIZE              64
#define NUM_ENDPOINTS         5
#define NUM_USB_BUFFERS 30
```

```
#define NUM_INTERFACE           3
#define CDC_IAD_DESCRIPTOR      1
#define CDC_STATUS_INTERFACE    0
#define CDC_DATA_INTERFACE      1       // Ser
#define CDC_ACM_ENDPOINT        1
#define CDC_RX_ENDPOINT       2
#define CDC_TX_ENDPOINT       3
#define CDC_ACM_SIZE          16
#define CDC_RX_SIZE           64
#define CDC_TX_SIZE           64
#define MIDI_INTERFACE        2 // MIDI
#define MIDI_TX_ENDPOINT      4
#define MIDI_TX_SIZE          64
#define MIDI_RX_ENDPOINT      5
#define MIDI_RX_SIZE          64
#define ENDPOINT1_CONFIG        ENDPOINT_TRANS
#define ENDPOINT2_CONFIG        ENDPOINT_RECEI
#define ENDPOINT3_CONFIG        ENDPOINT_TRANS
#define ENDPOINT4_CONFIG        ENDPOINT_TRANS
#define ENDPOINT5_CONFIG        ENDPOINT_RECEI
```

Reply With Quote

12-11-2015, 08:27 AM #16

sjtrny

Junior Member

Join Date: Dec 2015
Posts: 5

Maybe Paul can answer this: Why isn't this the default def for USB_MIDI?

Reply With Quote

02-13-2016, 05:12 PM #17

Pensive

Senior Member

Join Date: Aug 2014
Location: Basingstoke, UK
Posts: 562

> Originally Posted by **sjtrny**
> Maybe Paul can answer this: Why isn't this the default def for USB_MIDI?

This is a good point - it would be good to work on this.

Or indeed include standard midi function that had a sysex println solution which works with midi-ox.

I'm working on it now to see if it's possible and will submit a pull if I can get it to work.

Reply With Quote

02-13-2016, 05:34 PM #18

Pensive

Senior Member

Join Date: Aug 2014
Location: Basingstoke, UK
Posts: 562

Hangon thats interesting.

I'm using visual micro and it's not working with Teensy emulated serial - that's my problem. Serial output is working fins in the arduino IDE.

Going to keep it simple and use arduino IDE monitor for now and report the issue on visualmicro forum.

Reply With Quote

02-15-2016, 08:56 PM #19

Pensive

Senior Member

Join Date: Aug 2014
Location: Basingstoke, UK
Posts: 562

> Originally Posted by **Pensive**
> Going to keep report the issue on visualmicro forum.

Link here for anyone who is interested.

Reply With Quote

04-13-2017, 07:33 PM #20

silbo

Junior Member

Join Date: Apr 2017
Posts: 1

Hey, so using MIDI + Serial mode allows only the Teensy to send Serial data using the Serial.print() ?
It looks that way anyway, I was unable to process incoming serial data on Teensy. So I have to go the same way, just using MIDI messaged to communicated to the Teensy, System exclusive messages.

Reply With Quote

Quick Navigation **Project Guidance** **Top**

« Previous Thread | Next Thread »

Tags for this Thread

midi, serial, time, type, usb
View Tag Cloud

Posting Permissions

You may not post new threads
You may not post replies
You may not post attachments
You may not edit your posts

BB code is On
Smilies are On
[IMG] code is On
[VIDEO] code is On
HTML code is Off

Forum Rules