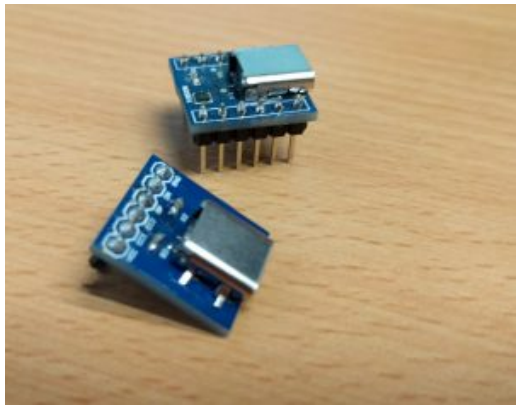
Using USB Type-C on hobyist projects

By Tyler | March 17, 2016 23 Comments

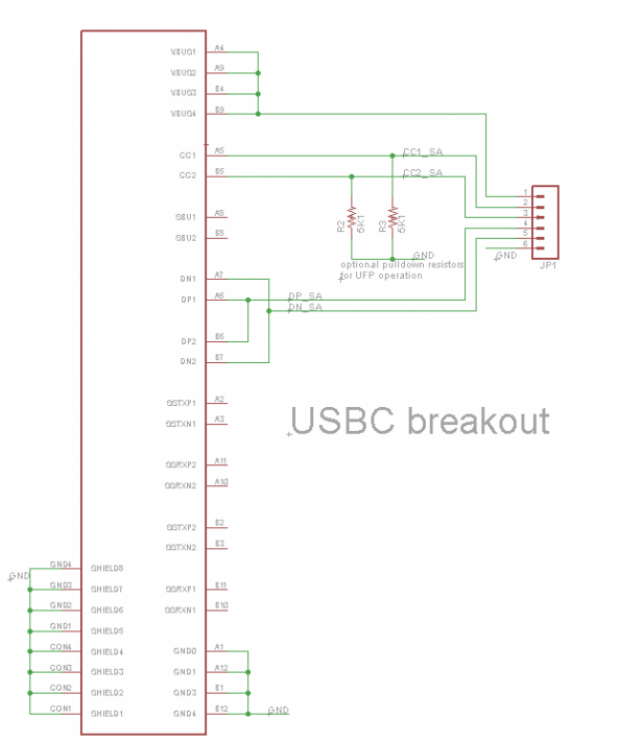
The new Type-C USB connector is the latest addition to the USB connector standards. It offers reversible plugs, direction independent cables, USB3.1 speeds, and 3A charging in a connector only a little bigger than the USB 2.0 MicroB connector. In order to add these capabilities the plugs and connectors have additional configuration pins to allow devices to negotiate their state. Supporting the configuration channel may seem like a difficult challenge but it can be achieved fairly simply for the basic use cases.



USB-Type C breakout boards

In this mode the Type-C device is acting as an UFP (Upstream facing port).
This is the equivalent of devices equipped with either a B, Mini-B or Micro-B socket and majority of hobbyist projects will probably fit into this category.
Luckily this is fairly easy to implement as we can avoid worrying about multiplexing or managing connection states.

The simplest implementation of this is to use two 5.1k pull down resistors on the CC lines.
This allows for detection by other devices when using a USB C-C cable.
The USB 2.0 Data lines can both be connected together at the receptacle as only one pair will be populated in the connector.
The power and ground lines will all want connecting to the appropriate power rails on your board.
The Sideband pins and superspeed pins should be left unconnected in this configuration.



USB-C breakout schematic

If you wish to have a device plug directly into a Type-C receptacle this can be done much the same as above with a few small differences.
As the orientation of the plug is fixed at your end only one pull-down configuration resistor (5.1k) will be needed (the one attached to A5). This needs to be the correct cc channel as some devices multiplex the USB2.0 data lines.
In addition only one set of USB 2 data pins should be present in the plug (A6 and A7) so there isn't a need to connect to both pairs like the previous case.

Using Type-C power on devices

If you want to use Type-C power which provides up to 3A at 5V (note: this is different to the USB Power Delivery capabilities) you will need to monitor the voltages at the CC lines to perform power detection. This can be achieved by either using an ADC to measure the CC lines (or using a IC such as the one discussed further down to manage the CC detection for you). Table 4-36 in the USB Spec (version 2.0) covers the voltages that represent the power sources capabilities. note: the capabilities of a host may change while connected so this needs to be monitored throughout the connection.

Table 4-36 Voltage on Sink CC pins (Multiple Source Current Advertisements)

Detection	Min voltage	Max voltage	Threshold
vRa	−0.25 V	0.15 V	0.2 V
vRd-Connect	0.25 V	2.04 V	
vRd-USB	0.25 V	0.61 V	0.66 V
vRd-1.5	0.70 V	1.16 V	1.23 V
vRd-3.0	1.31 V	2.04 V	

Excerpt from the Type-C spec (Table 4-36)

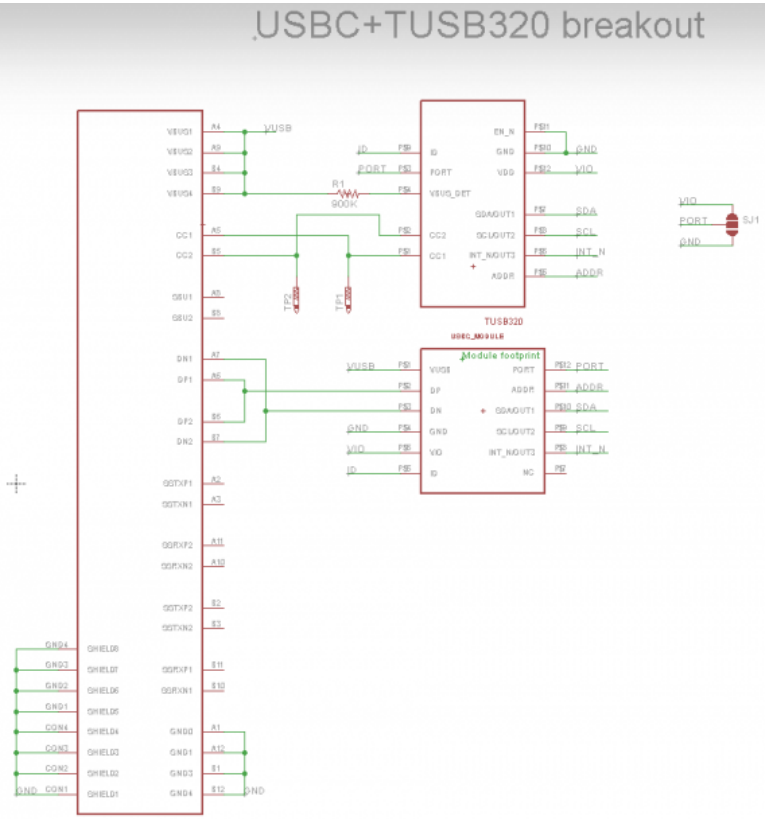
This can be simplified into the folowing states of a CC line

- less than 0.2v nothing is plugged in/the other CC line is connected.
- between 0.2v and 0.66v only default USB power is available.
- between 0.66v and 1.23v 1.5A USB-C power is available.
- above 1.23v 3.0A USB-C power is available.

Type-C as a 2.0 host or dual role device.

Using Type-C as a DFP (downstream facing port) or a DRP (dual role port) is a bit more complicated than the previous example.
This is because the host needs to detect the connection before connecting VBUS to the device.

While it would be possible to do this yourself there are configuration channel control ICs available to perform this task such as the TUSB320.
These ICs handle the majority of the work for you.
The bits you are left with are turning on VBUS once connection has been established and putting your device into the correct USB mode.
This is often already provided by your controller if it supports USB on-the-go, many configuration ICs have an ID pin that acts like the ID pin on on the go connector for this reason.
Even if you are using the device in host only mode, you will need to enable VBUS only when a connection has been established to prevent two USB devices attempting to provide power at the same time.
An example schematic using the TUSB320 is shown below.



USB-C breakout schematic with control IC

USB 3.1 operation and USB Power Delivery

While USB3.1 superspeed multiplexing and USB power delivery are outside of the scope of this guide there are ICs available which provide these functionalities however they are often in BGA packages which are difficult to work with.

References

- Hacaday.io Project

Updates

- 2019/11/09: Clarified Type C current section and fixed references as table number have changed in newer specification versions

Category: Guide Tags: guide, USB, USB Type-C

23 thoughts on “Using USB Type-C on hobyist projects”

Pingback: DIY USB Type C | Hackaday

Benson Leung
May 19, 2016

Hi Tyler!

Great guide! One quick minor correction I noticed in your description of Type-C UFPs.

“The simplest implementation of this is to use two 56k pull down resistors on the CC lines.
This allows for detection by other devices when using a USB C-C cable.”

The Rd pulldown should be a 5.1kΩ resistor. The schematic looks correct though with a 5K1 resistor. Otherwise, thanks for posting this! It should be invaluable for hobbyists looking to use Type-C in their designs!

Tyler Post author
May 19, 2016

Hi Benson

Thanks for spotting that, I have updated the article

Tyler

Pingback: Using USB Type-C on hobbyist projects - Electronics-Lab

Bryan McGrew
May 25, 2016

Hi! I'm still new at electronics tinkering and would like any help you could give me on USB C. I want to make a custom PiHat that would allow me to add at least one usb type c to my raspberry pi. I'd also like to be able to make my own alternate mode (if possible) that would essentially allow me to access the pi's gpio. Thank you in advance.

Tyler Post author
May 26, 2016

Hi Bryan

Tyler

Bryan McGrew
May 27, 2016

Thank you so much!

Henri Bergius
March 27, 2017

Hi! Thanks for posting this!

Where on the USB 2.0 side should the CC1 and CC2 lines connect? That is a bit unclear from the JP1 representation.

Trying to use this to add a USB-C plug to a microcontroller.

Tyler Post author
March 27, 2017

Hi Henri

Jonathan
May 16, 2017

Hi!

Sorry for being a total jackass here but could you please assist me with something that I can't figure out?

If I want to convert a micro USB port to USB C (for a DIY project of mine).
How should i wire it up to be able to use the USB 2.0 device both with a USB type A (Master) > USB-C (Slave) and USB C (Master) > USB C (Slave)?

In other words, I want it to be compatible with both my desktop PC (USB A) and my laptop (USB C).

Kind regards,
Jonathan

Tyler Post author
May 17, 2017

Hi Jonathan

You will want to have a type C receptacle on your device with the two pull down resistors. See the "Type-C as a 2.0 device with a type-C receptacle" section for a schematic and details.
You can then use an A-C cable when using it with your desktop and a C-C cable when using with your laptop.

Alex Lomis
February 15, 2018

Hi, I'm having some issues when I follow the instructions for "Type-C as a 2.0 device with a type-C receptacle." I've wired it up how the schematic shows (using 4.7kOhm resistors instead of 5.1k), yet when I plug it in to windows 10 I get an error saying "the last USB device you plugged into this computer malfunctioned…"

I've checked my wiring many times, and still have had no luck with getting it to work. I'm trying to put a USB C connector an an Arduino Nano.

Any help would be greatly appreciated!
-Alex

Tyler Post author
February 16, 2018

Hi Alex

I have had that error a few times on windows, I have seen it when there is a connection problem with the USB data lines or configuration on the microcontroler such as a bad USB descriptor.

Some things that you could help in debugging it.
Check for signal continuity through the connector and that the data lines aren't shorted to anything else (the C connectors are quite fine pitch so this can happen fairly easily) I tend to use a USb breakout on the end of a cable for testing but you can get a multimeter probe into a USB A connector if you

In the case of USB descriptor problems I have found the linux lsusb command or the USBView utility from Microsoft to be useful in checking if it is a low level signal problem or a problem with the device firmware, usually devices with signal problems wont enumerate atall.
I hope that is useful in getting it fixed.

Tyler

quentin
March 6, 2019

Hey Tyler, I wanted to use your diagrams to convert a micro USB teensy to USB-C (as an UFP).

You used to have great wiring diagrams, any chance you still have them somewhere ?

Many thanks !

Tyler Post author
June 27, 2019

Hi Quentin

Apologies for this, looks like they got misplaced during the server move, they are back up now, glad they are usefull.

Thanks

Restel
March 21, 2019

Hey.
The images in the post have seem to become 404

Matt Wolfe
November 5, 2019

Thank you for posting this! I would like to power a dumb device from a USB-C port and I only need 500mA @5V (USB 2.0 default power). Do I need to do anything more than the pair of 5.1k pull downs on the CC pins to keep the power flowing from the source?

Tyler Post author
November 7, 2019

Hi Matt

That is correct, for a minimal power only implementation as long as you have the cc puldown resistors then you will be able to get power from an upstream device

Hi, Tyler.
Sorry to dig up this older article, but I'm struggling a little. I'm making an adapter to power a device that requires 5V and can draw up to 3A. The adapter will simply have a USB C receptacle to use a charger as the source of power. If I understand it correctly, placing a 5.1K pull down resistor on both CC1 and CC2 will make the adapter look like a USB 2.0 device and, i think, only make 500mA available. What do I need to do on the adapter (USB C receptacle) end to ensure 3A @ 5V is made available by the charger?

Thanks,
oz

Tyler Post author
November 9, 2019

Hi Oz

The pulldown resistors indicate to an upstream device that a client is connected and will be the same regardless how much power a device uses. Chargers or USB-C host devices instead indicate the amount of power they can provide using differing pullup resistors on the other end of the CC lines. To detect the capabilities of the carger you will need to sense the voltage on the CC pins. I have updated the section above on Type-C current detection to include the table from the specification (the table number has changed in recent versions so no longer linked to the correct one) which shows what voltages are present on the CC pin for each current capability. Once you have sensed the CC lines you can tell what current is avaliable from the port and how much power you are able to draw. You might be connected to a device which can't supply 3A type-C current, your options when this happens are to either run at a lower power level or only fully power on if the full 3A is avaliable.

Hope that helps

Tyler

Tyler,
That makes so much sense. I basically had the responsibility flipped. It was starting to dawn on me as I continued to poke around that it was up to the device to not draw more current than it should. I would like to make a responsible device that doesn't turn on unless it senses it can draw up to 3A. Do you have any recommendations for ICs that can handle that detection? Maybe I'll break out LTSpice and figure out how to toggle my power protection chip's enable pin using basic components… My EE skills are rusty.

Thanks,
oz

oz
November 9, 2019

Just re-read the post. TUSB320! Time to read some datasheets.
Thanks for all your help!
oz

Tyler Post author
November 10, 2019

As well as chips like the TUSB320 (combined with an external logic gate) you can also use a pair of analogue input pins on a microcontroller to read the values from the CC lines and then control your power supplies enable pin. Using more basic components a pair of analogue comparators and logic to connect their outputs would also work, but the other options are easier to do.