
2024-01-06 usb on the go

USB, the Universal Serial Bus, was first released in 1996. It did not achieve widespread adoption until some years later; for most of the '90s RS-232-ish serial and its awkward sibling the parallel port were the norm for external peripheral. It's sort of surprising that USB didn't take off faster, considering the significant advantages it had over conventional serial. Most significantly, USB was self-configuring: when you plugged a device into a host, a negotiation was performed to detect a configuration supported by both ends. No more decoding labels like 9600 8N1 and then trying both flow control modes!

There are some significant architectural differences between USB and conventional serial that come out of autoconfiguration. Serial ports had no real sense of which end was which. Terms like DTE and DCE were sometimes used, but they were a holdover from the far more prescriptive genuine RS-232 standard (which PCs and most peripherals did not follow) and often inconsistently applied by manufacturers. All that really mattered to a serial connection is that one device's TX pin went to the other device's RX pin, and vice versa. The real differentiation between DCE and DTE was the placement of these pins: in principle, a computer would have them one way around, and a peripheral the other way around. This meant that a straight-through cable would result in a crossed-over configuration, as expected.

In practice, plenty of peripherals used the same DE-9 wiring convention as PCs, and sometimes you wanted to connect two PCs to each other. Some peripherals used 8p8c modular jacks, some peripherals used real RS-232 connectors, and some peripherals used monstrosities that could only have emerged from the nightmares of their creators. The TX pin often ended up connected to the TX pin and vice versa. This did not work. The solution, as we so often see in networking, was a special cable that crossed over the TX and RX wires within the cable (or adapter). For historical reasons this was referred to as a null modem cable.

One of the other things that was not well standardized with serial connections was the gender of the connectors. Even when both ends features the PC-standard DE-9, there was some inconsistency over the gender of the connectors on the devices and on the cable. Most people who interact with serial with any regularity probably have a small assortment of "gender changers" and null-modem shims in their junk drawer. Sometimes you can figure out the correct configuration from device manuals (the best manuals provide a full pinout), but often you end up guessing, stringing together adapters until the genders fit and then trying with and without a null modem adapter.

You will notice that we rarely go through this exercise today. For that we can thank USB's very prescriptive standards for connectors on devices and cables. The USB standard specifies three basic connectors, A, B, and C. There are variants of some connectors, mostly for size (mini-B, micro-B, even a less commonly used mini-A and micro-A). For the moment, we will ignore C, which came along later and massively complicated the situation. Until 2014, there was only A and B. Hosts had A, and devices had B.

Yes, USB fundamentally employs a host-device architecture. When you connect two things with

USB, one is the host, and the other is the device. This differentiation is important, not just for the cable, but for the protocol itself. USB prior to 3, for example, does not feature interrupts. The host must poll the device for new data. The host also has responsibility for enumeration of devices to facilitate autoconfiguration, and for flow control throughout a tree of USB devices.

This architecture makes perfect sense for USB's original 1990s use-case of connecting peripherals (like mice) to hosts (like PCs). In fact, it worked so well that once USB1.1 addressed some key limitations it became completely ubiquitous. Microsoft used the term "legacy-free PC" to identify a new generation of PCs at the very end of the '90s and early '00s. While there were multiple criteria for the designation, the most visible to users was the elimination of multiple traditional ports (like the game port! remember those!) in favor of USB.

Times change, and so do interconnects. The consumer electronics industry made leaps and bounds during the '00s and "peripheral" devices became increasingly sophisticated. The introduction of portables running sophisticated operating systems pushed the host-device model to a breaking point. It is, of course, tempting to talk about this revolution in the context of the iPhone. I never had an iPhone though, so the history of the iDevice doesn't have quite the romance to me that it has to so many in this space [1]. Instead, let's talk about Nokia. If there is a Windows XP to Apple's iPhone, it's probably Nokia. They tried so hard, and got so far, but [...].

The Nokia 770 Internet Tablet was not by any means the first tablet computer, but it was definitely a notable early example. Introduced in 2005, it premiered the Linux-based Maemo operating system beloved by Nokia fans until iOS and Android killed it off in the 2010s. The N770 was one of the first devices to fall into a new niche: with a 4" touchscreen and OMAP/ARM SoC, it wasn't exactly a "computer" in the consumer sense. It was more like a peripheral, something that you would connect to your computer in order to load it up with your favorite MP3s. But it also ran a complete general-purpose operating system. The software was perfectly capable of using peripherals itself, and MP3s were big when you were storing them on MMC. Shouldn't you be able to connect your N770 to a USB storage device and nominate even more MP3s as favorites?

Obviously Linux had mainline USB mass storage support in 2005, and by extension Maemo did. The problem was USB itself. The most common use case for USB on the N770 was as a peripheral, and so it featured a type-B device connector. It was not permitted to act as a host. In fact, every PDA/tablet/smartphone type device with sophisticated enough software to support USB peripherals would encounter the exact same problem. Fortunately, it was addressed by a supplement to the USB 2.0 specification released in 2001.

The N770 did not follow the supplement. That makes it fun to talk about, both because it is weird and because it is an illustrative example of the problems that need to be solved.

The N770 featured an unusual USB transceiver on its SoC, seemingly unique to Nokia and called "Tahvo." The Tahvo controller exposed an interface (via sysfs in the Linux driver) that allowed the system to toggle it between device mode (its normal configuration) and host mode. This worked well enough with Maemo's user interface, but host mode had a major limitation. The N770 wouldn't provide power on the USB port; it didn't have the necessary electrical components. Instead, a special adapter cable was needed to provide 5v power from an alternate source.

So there are several challenges for a USB device to operate as host or device:

- The USB controller needs a way to determine if it should behave in host or device mode. Ideally, the user shouldn't have to think about this.

- The USB controller needs to be able to supply power when in host mode, and in most practical situations also needs to accept power (e.g. for charging) when in device mode.

Note that "special cable" involved in host mode for the N770. You might think this was the ugliest part of the situation. You're not wrong, but it's also not really the hack. For many years to follow, the proper solution to this problem would also involve a special cable.

As I mentioned, since 2001 there has been a supplement USB specification called USB On-The-Go, commonly referred to as USB OTG, perhaps because On-The-Go is an insufferably early '00s name. It reminds me of, okay, here goes a full-on anecdote.

Anecdote

I attended an alternative middle school in Portland that is today known as the Sunnyside Environmental School. I could tell any number of stories about the bizarre goings-on at this school that you would scarcely believe, but it also had its merits. One of them, which I think actually came from the broader school district, was a program in which eighth graders were encouraged to "job shadow" someone in a profession they were interested in pursuing. By good fortune, a friend's father was an electrical engineer employed at Intel's Jones Farm campus, and agreed to be my host. I had actually been to Jones Farm a number of times on account of various extracurricular programs (in that era, essentially every STEM program in the Pacific Northwest operated on the largess of either Intel or Boeing, if not both). This was different, though: this guy had a row of engraved brass patent awards lining his cubicle wall and showed me through labs where technicians tinkered with prototype hardware. Foreshadowing a concerning later trend in my career, though, the part that stuck with me most was the meetings. I attended meetings, including one where this engineering team was reporting to leadership on the status of a few of their projects.

I am no doubt primed to make this comparison by the mediocre movie I watched last night, but I have to describe the experience as Wonka-esque. These EEs demonstrated a series of magical hardware prototypes to some partners from another company. Each was more impressive than the last. It felt like I was seeing the future in the making.

My host demonstrated his pet project, a bar that contained an array of microphones and used DSP methods to compare the audio from each and directionalize the source of sounds. This could be used for a sophisticated form of noise canceling in which sound coming from an off-axis direction could be subtracted, leaving only the voice of the speaker. If this sounds sort of unremarkable, that is perhaps a reflection of its success, as the same basic concept is now implemented in just about every laptop on the market. Back then, when the N770 was a new release, it was challenging to make work and my host explained that the software behind it usually crashed before he finished the demo, and sometimes it turned the output into a high pitched whine and he hadn't quite figured out why yet. I suppose that meeting was lucky.

But that's an aside. A long presentation, and then debate skeptical execs, revolved around a new generation of ultramobile devices that Intel envisioned. One, which I got to handle a prototype of, would eventually become the Intel Core Medical Tablet. It featured chunky, colorful design that is clearly of the same vintage as the OLPC. It was durable enough to stand on, which a lab technician demonstrated with delight (my host, I suspect tired of this feat, picked up some sort of lab interface and dryly remarked that he could probably stand on it too). The Core Medical Tablet shared another trait with the OLPC: the kind of failure that leaves no impact on the world but a big footprint at recyclers. Years later, as an intern at Free Geek, I would come across at least a dozen.

Another facet of this program, though, was the Mobile Metro. The Metro was a new category of subnotebook, not just small but thin. A period article compares its 18mm profile to the somewhat thinner Motorola Razr, another product with an outsize representation in the Free Geek Thrift Store. Intel staff were confident that it would appeal to a new mobile workforce, road warriors working from cars and coffee shops. The Mobile Metro featured SideShow, a small e-ink display (in fact, I believe, a full Windows Mobile system) on the outside of a case that could show notifications and media controls.

The Mobile Metro was developed around the same time as the Classmate PC, but seems to have been even less successful. It was still in the conceptual stages when I heard of it. It was announced, to great fanfare, in 2007. I don't think it ever went into production. It had WiMax. It had inductive charging. It only had one USB port. It was, in retrospect, prescient in many ways both good and bad.

The point of this anecdote, besides digging up middle school memories while attempting to keep others well suppressed, is that the mid-2000s were an unsettled time in mobile computing. The technology was starting to enable practical compact devices, but manufacturers weren't really sure how people would use them. Some innovations were hits (thin form factors). Some were absolute misses (SideShow). Some we got stuck with (not enough USB ports).

End of anecdote

As far as I can tell, USB OTG wasn't common on devices until it started to appear on Android smartphones in the early 2010s. Android gained OTG support in 3.1 (codenamed Honeycomb, 2011), and it quickly appeared in higher-end devices. Now OTG support seems nearly universal for Android devices; I'm sure there are lower-end products where it doesn't work but I haven't yet encountered one. Android OTG support is even admirably complete. If you have an Android phone, amuse yourself sometime by plugging a hub into it, and then a keyboard and mouse. Android support for desktop input peripherals is actually very good and operating mobile apps with an MX Pro mouse is an entertaining and somewhat surreal experience. On the second smartphone I owned, I hazily think a Samsung in 2012-2013, I used to take notes with a USB keyboard.

iOS doesn't seem to have sprouted user-exposed OTG support until the iPhone 12, although it seems like earlier versions probably had hardware support that wasn't exposed by the OS. I could be wrong about this; I can't find a straightforward answer in Apple documentation. The Apple Community Forums seem to be... I'll just say "below average." iPads seem to have gotten OTG support a lot earlier than the iPhone despite using the same connector, making the situation rather confusing. This comports with my general understanding of iOS, though, from working with bluetooth devices: Apple is very conservative about hardware peripheral support in iOS, and so it's typical for iOS to be well behind Android in this regard for purely software reasons. Ask me about how this has impacted the Point of Sale market. It's not positive.

But how does OTG work? Remember, USB specifies that hosts must have an A connector, and devices a B connector. Most smartphones, besides Apple products and before USB-C, sported a micro-B connector as expected. How OTG?

The OTG specification decouples, to some extent, the roles of A/B connector, power supply, and host/device role. A device with USB OTG support should feature a type AB socket that accommodates either an A or a B plug. Type AB is only defined for the mini and micro sizes, typically used on portable devices. The A or B connectors are differentiated not only by the shape of their shells (preventing a type-A plug being inserted into a B-only socket), but also electrically. The observant among you may have noticed that mini and micro B sockets and plugs feature five pins, while USB2.0 only uses four. This is the purpose of the fifth pin:

differentiation of type A and B plugs.

In a mini or micro type B plug, the fifth pin is floating (disconnected). In a mini or micro type A plug, it is connected to the ground pin. When you insert a plug into a type AB socket, the controller checks for connectivity between the fifth pin (called the ID pin) and the ground. If connectivity is present, the controller knows that it must act as an OTG A-device---it is on the "A" end of the connection. If there is no continuity, the more common case, the controller will act as an OTG B-device, a plain old USB device [2].

The OTG A-device is always responsible for supplying 5v power (see exception in [2]). By default, the A-device also acts as the host. This provides a basically complete solution for the most common OTG use-case: connecting a peripheral like a flash drive to your phone. The connector you plug into your phone identifies itself as an A connector via the ID pin, and your phone thus knows that it must supply power and act as host. The flash drive doesn't need to know anything about this, it has a B connection and acts as a device as usual. This simple case only became confusing when you consider a few flash drives sold specifically for use with phones that had a micro-A connector right on them. These were weird and I don't like them.

In the more common situation, though, you would use a dongle: a special cable. A typical OTG cable, which were actually included in the package with enough Android phones of the era that I have a couple in a drawer without having ever purchased one, provides a micro-A connector on one end and a full-size A socket on the other. With this adapter, you can plug any USB device into your phone with a standard USB cable.

Here's an odd case, though. What if you plug two OTG devices into each other? USB has always had this sort of odd edge-case. Some of you may remember "USB link cables," which don't really have a technical name but tend to get called Laplink cables after a popular vendor. Best Buy and Circuit City used to be lousy with these things, mostly marketed to people who had bought a new computer and wanted to transfer their files. A special USB cable had two A connectors, which might create the appearance that it connected two hosts, but in fact the cable (usually a chunky bit in the middle) acted as two devices to connect to two different hosts. The details of how these actually worked varied from product to product, but the short version is "it was proprietary." Most of them didn't work unless you found the software that came with them, but there are some pseudo-standard controllers supported out of the box by Windows or Linux. I would strongly suggest that you protect your mental state by not trying to use one.

OTG set out to address this problem more completely. First, it's important to understand that this in no way poses an exception to the rule that a USB connection has an A end and a B end. A USB cable you use to connect two phones together might, at first glance, appear to be B-B. But, if you inspect closer, you will find that one end is mini or micro A, and the other is mini or micro B. You may have to look close, the micro connectors in particular have a similar shell!

If you are anything like me, you are most likely to have encountered such a cable in the box with a TI-84+. These calculators had a type AB connector and came with a micro A->B cable to link two units. You might think, by extension, that the TI-84+ used USB OTG. The answer is kind of! The USB implementation on the TI-84+ and TI-84+SE was very weird, and the OS didn't support anything other than TIConnect. Eventually the TI-84+CE introduced a much more standard USB controller, although I think support for any OTG peripheral still has to be hacked on to the OS. TI has always been at the forefront of calculator networking, and it has always been very weird and rarely used.

This solves part of the problem: it is clear, when you connect two phones, which should supply power and which should handle enumeration. The A-device is, by default, in charge. There are problems where this interacts with common USB devices types, though. One of the most common uses of USB with phones is mass storage (and its evil twin MTP). USB mass storage has a very

strong sense of host and device at a logical level; the host can browse the devices files. When connecting two smartphones, though, you might want to browse from either end. Another common problem case here is that of the printer, or at least it would be if printer USB host support was ever usable. If you plug a printer into a phone, you might want to browse the phone as mass storage on the printer. Or you might want to use conventional USB printing to print a document from the phone's interface. In fact you almost certainly want to do the latter, because even with Android's extremely half-assed print spooler it's probably a lot more usable than the file browser your printer vendor managed to offer on its 2" resistive touchscreen.

OTG adds Host Negotiation Protocol, or HNP, to help in this situation. HNP allows the devices on a USB OTG connection to swap roles. While the A-device will always be the host when first connected, HNP can reverse the logical roles on demand.

This all sounds great, so where does it fall apart? Well, the usual places. Android devices often went a little off the script with their OTG implementations. First, the specification did not require devices to be capable of powering the bus, and phones couldn't. Fortunately that seems to have been a pretty short lived problem, only common in the first couple of generations of OTG devices. This wasn't the only limitation of OTG implementations; I don't have a good sense of scale but I've seen multiple reports that many OTG devices in the wild didn't actually support HNP, they just determined a role when connected based on the ID pin and could not change after that point.

Finally, and more insidiously, the whole thing about OTG devices having an AB connector didn't go over as well as intended. We actually must admire TI for their rare dedication to standards compliance. A lot of Android phones with OTG support had a micro-B connector only, and as a result a lot of OTG adapters use a micro-B connector.

There's a reason this was common; since A and B plugs are electrically differentiable regardless of the shape of the shell, the shell shape arguably doesn't matter. You could be a heavy OTG user with such a noncompliant phone and adapter and never notice. The problem only emerges when you get a (rare) standards-compliant OTG adapter or, probably more common, OTG A-B cable. Despite being electrically compatible, the connector won't fit into your phone. Of course this behavior feeds itself; as soon as devices with an improper B port were common, manufacturers of cables were greatly discouraged from using the correct A connector.

The downside, conceptually, is that you could plug an OTG A connector (with a B-shaped shell) into a device with no OTG support. In theory this could cause problems, in practice the problems don't seem to have been common since both devices would think they were B devices and (if standards compliant) not provide power. Essentially these improper OTG adapters create a B-B cable. It's a similar problem to an A-A cable but, in practice, less severe. Like an extension cord with two female ends. Home Depot might even help you make one of those.

While trying to figure out which iPhones had OTG support, I ran across an Apple Community thread where someone helpfully replied "I haven't heard of OTG in over a decade." Well, it's not a very helpful reply, but it's not exactly wrong either. No doubt the dearth of information on iOS OTG is in part because no one ever really cared. Much like the HDMI-over-USB support that a generation of Android phones included, OTG was an obscure feature. I'm not sure I have ever, even once, seen a human being other than myself make use of OTG.

Besides, it was completely buried by USB-C.

The thing is that OTG is not gone at all, in fact, it's probably more popular than ever before. There seems to be some confusion about how OTG has evolved with USB specifications. I came across more than one article saying that USB 3.1 Dual Role replaced OTG. This assertion is... confusing. It's not incorrect, but there's a good chance of it leading you int he wrong

direction.

Much of the confusion comes from the fact that Dual-Role doesn't mean anything that specific. The term Dual-Role and various resulting acronyms like DRD and DRP have been applied to multiple concepts over the life of USB. Some vendors say "static dual role" to refer to devices that can be configured as either host or device (like the N770). Some vendors use dual role to identify chipsets that detect role based on the ID pin but are not actually capable of OTG protocols like HNP. Some articles use dual role to identify chipsets with OTG support. Subjectively, I think the intent of the changes in USB 3.1 were mostly to formally adopt the "dual role" term that was already the norm in informal use---and hopefully standardize the meaning.

For USB-C connectors, it's more complicated. USB-C cables are symmetric, they do not identify a host or device end in any way. Instead, the USB-C ports use resistance values to indicate their type. When either end indicates that it is only capable of the device role, the situation is simple, behaving basically the same way that OTG did: the host detects that the other end is a device and behaves as the host.

When both ends support the host role, things work differently: the Dual Role feature of USB-C comes into play. The actual implementation is reasonably simple; a dual-role USB-C controller will attempt to set up a connection both ways and go with whichever succeeds. There are some minor complications on top of this, for example, the controller can be configured with a "preference" for host or device role. This means that when you plug your phone into your computer via USB-C, the computer will assume the host role, because although it's capable of either the phone is configured with a preference for the device role. That matches consumer expectations. When both devices are capable of dual roles and neither specifies a preference, the outcome is random. This scenario is interesting but not all that common in practice.

The detection of host or device role by USB-C is based on the CC pins, basically a more flexible version of OTG's ID pin. There's another important difference between the behavior of USB-C and A/B: USB-C interfaces provide no power until they detect, via the CC pins, that the other device expects it. This is an important ingredient to mitigate the problem with A-A cables, that both devices will attempt to power the same bus.

The USB-C approach of using CC pins and having dual role controllers attempt one or the other at their preference is, for the most part, a much more elegant approach. There are a couple of oddities. First, in practice cables from C to A or B connectors are extremely common. These cables must provide the appropriate values on the CC pins to allow the USB-C controller to correctly determine its role, both for data and power delivery.

Second, what about role reversal? For type A and B connectors, this is achieved via HNP, but HNP is not supported on USB-C. Application notes from several USB controller vendors explain that, oddly enough, the only way to perform role reversal with USB-C is to implement USB Power Delivery (PD) and use the PD negotiation protocol to change the source of power. In other words, while OTG allows reversing host and device roles independently of the bus power source, USB-C does not. The end supplying power is always the host end. This apparent limitation probably isn't that big of a deal, considering that the role reversal feature of OTG was reportedly seldom implemented.

That's a bit of a look into what happens when you plug two USB hosts into each other. Are you confused? Yeah, I'm a little confused too. The details vary, and a lot more based on the capabilities of the individual devices rather than the USB version in use. This has been the malaise of USB for a solid decade now, at least: the specification has become so expansive, with so many non-mandatory features, that it's a crapshoot what capabilities any given USB port actually has. The fact that USB-C supports a bevy of alternate modes like Thunderbolt and

HDMI only adds further confusion.

I sort of miss when the problem was just inappropriate micro-B connectors. Nonetheless, USB-C dual role support seems ubiquitous in modern smartphones, and that's the only place any of this ever really mattered. Most embedded devices still seem to prefer to just provide two USB ports: a host port and a device port. And no one ever uses the USB host support on their printer. It's absurd, no one ever would. Have you seen what HP thinks is a decent file browser? Good lord.

[1] My first smartphone was the HTC Thunderbolt. No one, not even me, will speak of that thing with nostalgia. It was pretty cool owning one of the first LTE devices on the market, though. There was no contention at all in areas with LTE service and I was getting 75+Mbps mobile tethering in 2011. Then everyone else had LTE too and the good times ended.

[2] There are actually several additional states defined by fixed resistances that tell the controller that it is the A-device but power will be supplied by the bus. These states were intended for Y-cables that allowed you to charge your phone from an external charger while using OTG. In this case neither device supplies power, the external charger does. The details of how this works are quite straightforward but will be confusing to keep adding as an exception, so I'm going to pretend the whole feature doesn't exist.