Universal serial bus (USB) Type-C offers a number of features, including a high level of flexibility and convenience to end users. System designers must choose available options carefully so that the overall system cost stays within a reasonable limit. Two choices that have the most impact on cost and complexity of the systems are native Type-C power of 15 W versus enhanced power capability and video support. This article discusses how to implement a USB Type-C port so that it will minimally impact an existing system.

Introduction

In the electronics industry, USB Type-C is in the mind of every system designer. This interface consolidates data, power and video into a single connector interface. It is also bringing a real opportunity to eliminate power barrel jack connector from new platforms. USB Type-C supports USB 2.0 and USB 3.1, and provides options for alternate (Alt) modes such as DisplayPort for video. USB Type-C introduces native power capability of 15 W and an enhanced capability of up to 100 W with the addition of USB Power Delivery (USB PD). The interface introduces smaller, thinner and more robust connectors that are ready for data rates up to 20 Gbps. The cable is reversible and flippable and connects a host or a client device in either direction. System designers are thinking how to bring these desirable features and flexibility to their customers.

Let us consider an example where a system designer is implementing a new notebook platform. How much will incorporating a USB Type-C port add to the overall cost? How many new Type-C connectors are needed? Will all the connectors be full-featured? The flexibility and simplicity that USB Type-C offers to end users also adds complexity and cost to system implementations. While the new eco-system provides more options for implementation, system designers must tread the water carefully so that their overall system cost stays acceptable.

So what will the new laptop platform look like? Some system designer will probably choose to have one full-featured Type-C port that provides enhanced power charging with USB-PD. This super port will have Alt mode video. To save cost and complexity, a designer may want to choose other ports to have reduced features, such as native Type-C power of 15 W and USB data support.
A key consideration is implementing USB Type-C in place of legacy USB connectors while minimally disrupting an existing platform. This article outlines how to convert a USB 3.0 legacy port into a USB 3.1 Type-C port with minimal changes.

**Type-C USB 3.1 implementation**

USB Type-C has the same type of connectors at both ends – same receptacles at the host and client devices and the same plugs at both ends of the cable. Figure 1 shows a USB Type-C receptacle pin map. Note that the 24-pin interface is arranged in a symmetrical fashion that facilitates flipping the cable.

![USB Type-C receptacle pin map](Source: Type-C Specifications)

In addition to a USB 3.1 TX, RX, and USB 2.0 D+, D- signals, two CC pins are used for channel configuration (CC) and USB-PD communications. A typical system implementation shorts two D+ signals and two D- signals with stub connections, which eliminates the need for a USB 2.0 multiplexer (mux) to accommodate for a plug flip. However, such stubbed connections for USB 3.1 signals are not feasible due to signal integrity concerns requiring a 2:1 mux at each end of a Type-C interface. If Alt mode is used, a USB-PD function is required and the mux configuration becomes more complicated.

A typical USB 3.1 implementation includes two basic functions: a CC controller to manage the link; and a USB 3.1 mux for RX and TX signals to choose the connected side relative to Type-C plug orientation. The CC controller needs the ability to configure itself as a downstream-facing port (DFP), upstream-facing port (UFP) or dual-role port (DRP), depending on the desired system behavior. Table 1 summarizes the data/power behavior of different applications.
Table 1: Device data/power class for USB Type-C applications

<table>
<thead>
<tr>
<th>Data/power class</th>
<th>Example devices</th>
<th>Data/power mode/role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always source</td>
<td>Charger</td>
<td>DFP/source</td>
</tr>
<tr>
<td>Usually source</td>
<td>Laptop, battery bank</td>
<td>DFP/source with optional Try.SRC</td>
</tr>
<tr>
<td>Dual</td>
<td>Tablet</td>
<td>DFP/dual source and sink</td>
</tr>
<tr>
<td>Usually sink</td>
<td>Phone</td>
<td>UFP/sink with optional Try.SNK</td>
</tr>
<tr>
<td>Always sink</td>
<td>Portable drive, accessory</td>
<td>UFP/sink</td>
</tr>
</tbody>
</table>

Host-client implementation of USB Type-C

USB Type-C incorporates a channel configuration through which it establishes a USB link between a DFP and UFP. A DFP port can be considered as a host and UFP as a device in a traditional USB port definition. The CC function is used for the following determinations:

- DFP-to-UFP attach/detach detection and plug orientation
- DFP-to-UFP (host-to-device) and power relationship (provider/consumer) detection - without USB-PD by default a DFP (source) provides and a UFP (sink) consumes power
- USB Type-C VBUS current advertisement by provider and detection by consumer
- While attached, power and data roles can be changed only through USB-PD

Even though a receptacle has two CC pins, CC1 and CC2, only a single CC wire is connected through a cable. A DFP has a pull-up and a UFP has a pull-down for each CC pin. Monitoring the CC pins for specified voltage provides the orientation and attachment detection.
A DFP uses different resistor pull-up (or current source) values to advertise its current provider capability. Alternatively, a UFP detects how much current it can consume by applying a pull-down resistor and performing a voltage comparison. Three power settings are possible without USB-PD – legacy default (when enumerated, 900 mA for USB 3.1 and 500 mA for USB 2.0), 1.5A and 3A with 5V on $V_{BUS}$. 

**Figure 3** illustrates a typical host-client (DFP/UFP) implementation of USB Type-C. An example of a USB 3.1 host is a desktop or laptop PC. The Type-C port in a PC will likely be a DFP and act as a USB host and provide power for client devices. On the other hand, a typical example for a USB SuperSpeed client device can be a portable hard disk. The hard disk acts as a USB device and consumes power from $V_{BUS}$. 

---

**Figure 2: Channel configuration pull-up/pull-down model**
Figure 3: Typical host-client (DFP-UFP) implementation of USB Type-C

According to the Type-C specification, the responsibility of managing power consumption falls to the client/consumer. As a result, a client device is required to dynamically control power consumption based on a host’s current advertisement. An alternative is to keep the current consumption within the default limit. A DFP is likely to implement a current limit as well for added system protection.

If a DFP supports USB 3.1 it is required to provide 5V power using $V_{\text{CONN}}$ for active electronics inside a USB Type-C cable. $V_{\text{CONN}}$ is applied through the CC pin (CC1 or CC2) of the receptacle that is not connected via the CC wire through the cable. Instead, it powers circuits at the near-end plug. Note that every full-featured Type-C cable is required to have an electronic marker. Additionally, longer cables may need active signal conditioners.

**Dual-role port implementation of USB Type-C**

USB Type-C also defines a DRP that alternately identifies itself as DFP and UFP until a stable attached state is established. If a DRP is paired with a UFP or DFP, it takes the role of a DFP or UFP, respectively. If two DRPs are paired, the outcome is random, but can be influenced by two optional features: Try.SRC and Try.SNK. A DRP with a Try.SRC preference becomes a DFP (source), and with a Try.SNK preference it becomes a UFP (sink), if the other side has no preference. These features are important to have an orderly power provider/consumer relationship in the eco-system. For example, a laptop should charge a mobile phone – even when both have DRP capability.
An easy conversion of an existing USB platform with legacy connectors to USB Type-C without a major system re-design requires a CC controller device. For USB SuperSpeed support, an additional device with a USB SuperSpeed mux function is also needed.

The TUSB321 is an example of a single-chip USB Type-C port CC controller that can be configured as a DFP, UFP, or DRP. It is an autonomous device that virtually can be left alone and expected to work without intervention with some preset configurations. Software intervention is optional, but can provide additional functions that a system designer might find useful.
The HD3SS3212 is an example of a USB SuperSpeed passive mux that selects the active USB 3.1 signal pairs using a Signal Engineering Laboratories (SEL) signal provided by a CC controller to accommodate a Type-C plug-flip supporting USB 3.1 Gen 1 and Gen 2 for data rates of up to 10 Gbps.
Some systems may require signal boosting for USB SuperSpeed signals to pass compliance at the connector. A redriving mux can provide the dual function of signal conditioning and USB SuperSpeed switching. USB Type-C offers audio accessory features that can provide headphone and microphone function through Type-C connector, eliminating a 3.5 mm audio port in some systems. Audio signals use D+, D- and USB signals. To provide audio functions an additional mux will be required, which is not discussed in this article.

**Summary**

USB Type-C is expected to be hugely popular by electronics gadget lovers for its capability and flexibility. Systems do not have to incur significant development and component costs in order to provide its great benefits. For most basic implementation such as USB and 15 W power the conversion can be straightforward – you simply upgrade the connector and add a CC controller and an optional USB 3.1 SS mux.

**References**

1. Download a copy of the [USB Type-C specification](#).
2. Download these data sheets: [TUSB321](#), [HD3SS3212](#).
3. Download a [white paper](#) outlining USB2 only implementation.