∴ **ACRONAME**

Features

— Selectively connect one USB Type-C® (USB-C) connection to any one of 4 channels

— Bidirectional 1:4 or 4:1 configurations

— All ports support USB 3.2 Gen 2x1 link speeds up to 10Gbps[1]

— All ports support USB PD profiles up to 100W (20V, 5A)[2]

— Execute USB-C cable flip via software control[3]

— Supports pass through of USB Alt-Modes (DisplayPort, HDMI and Digital Audio)

— High Speed (HS) Data, SuperSpeed (SS) Data, CC/V_{CONN}, SBU, and V_{BUS} power can be independently enabled, disabled, or routed to any channel

— Measure V_{BUS}, V_{CONN} voltage and current on each channel

— Available in Passive or Redriver versions

— DIN-rail mountable

— Certified to withstand ±15kV ESD strikes (IEC61000-4-2 level 4)

Overview

The USB-C-Switch is a 4:1 software-programmable USB Type-C port selector and multiplexer switch, designed for demanding industrial environments where advanced control and monitoring of USB Type-C ports is required.

The USB-C-Switch can be used to selectively switch a USB connection from one common port to one of 4 mux ports, conduct Type-C cable flip operations, measure current and voltages on V_{BUS} and V_{CONN} lines of all ports, and independently control USB data and power connections on each port.

Typical applications include:

— Manufacturing testing of USB Type-C ports
— USB device validation and development
— USB functional testing
— USB peripheral management
— USB Alt-mode testing
— USB PD profile testing
— Regression test environments
— Automating USB Type C port "flip"
— Automating USB plug/unplug operations
— Automation of Apple CarPlay® or Android Auto® testing

Description

The USB-C-Switch gives engineers advanced control of USB connections in testing and development applications. The USB-C-Switch consists of several layers of internal switches to achieve the 4:1 selector and USB line control functionality. Without any hub or other directional intermediary devices, the USB-C switch is can behave "like a cable" to connected devices. USB2, USB3, power, CC, V_{conn}, and SBU, are passed through the USB-C-Switch between the common-port and the selected mux port. Data link, power negotiations and power between USB devices are provided by the attached devices themselves, allowing the USB-C-Switch to be used bidirectionally in either a 1:4 or 4:1 configuration.

Power and software control connections to the USB-C-Switch are established and maintained over a dedicated USB-C control port.

Each USB-C-Switch is uniquely addressable and controllable from a host PC via USB-C control port which also provide power. Built on top of Acroname's BrainStem® platform, the USB-C-Switch is easily controlled over USB with simple high-level APIs in C, C++, Python and LabVIEW.

---

[1] Passive version of S85-USBC-SWITCH may impose too much signal loss for systems to operate at 5Gbps or higher data rates; system link budget analysis required.

[2] USB Type-C 2.0 and Power Delivery 3.0

[3] Requires use of Acroname Universal Orientation Cable (UOC), C38-USBC-UOC

Passive and Redriver Models

There are two available models of the USB-C-Switch: passive and redriver. The two models have different hardware installed by Acroname during manufacturing; the model cannot be changed after delivery. Acroname places a label on the side of each USB-C-Switch indicating both the model and hardware revision (see example in Figure 1). High-level summary and intended applications of the two models are below with detailed differences in the specification tables.

Passive: Best for emulating off the shelf cables and for eye-diagram validation. Ordering part number: S85-PASS-USBCSW

Redriver: Best for general connectivity or longer cables. Includes a programmable, linear, equalizing redriver which allows USB signal tuning to compensate for insertion and cabling losses. Can provide a known-good reference transceiver plane for verification testing. Ordering part number: S85-RDVR-USBCSW
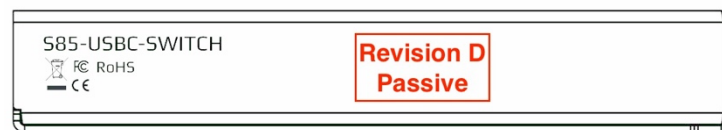

Figure 1: Side view showing model label location
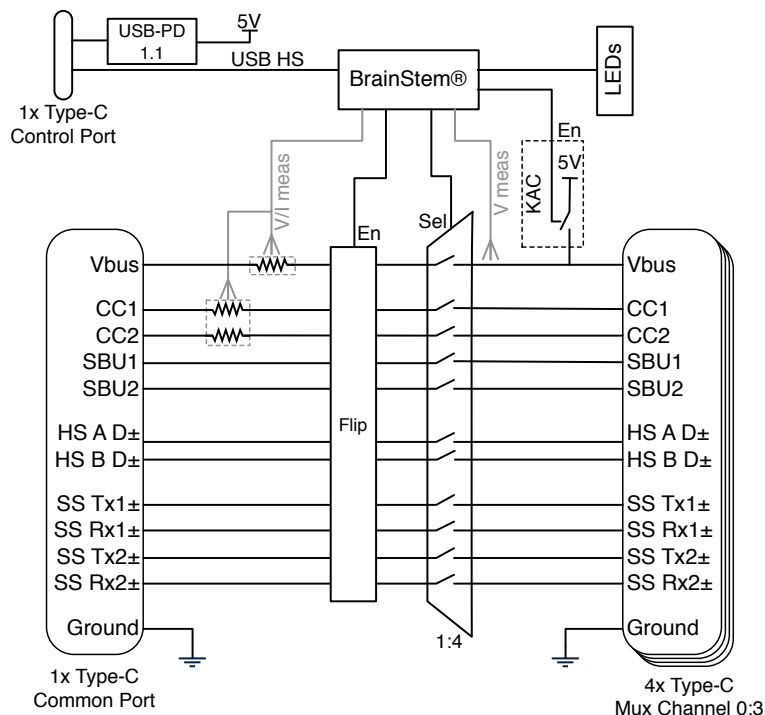
Passive Model Block Diagram

Figure 2: USB-C-Switch (Passive) functional block diagram for passive model

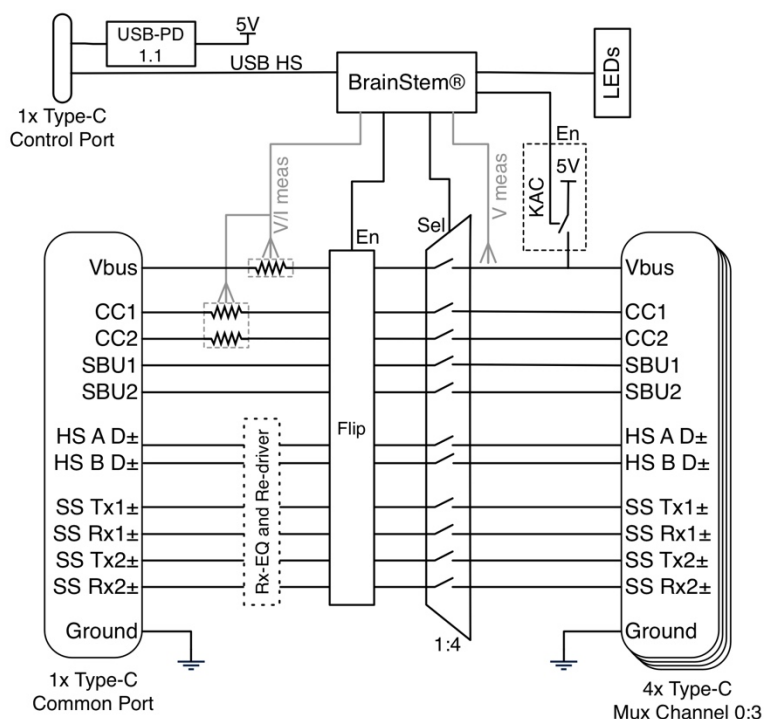# Redriver Model Block Diagram

Figure 3: USB-C-Switch functional block Diagram for redriver model
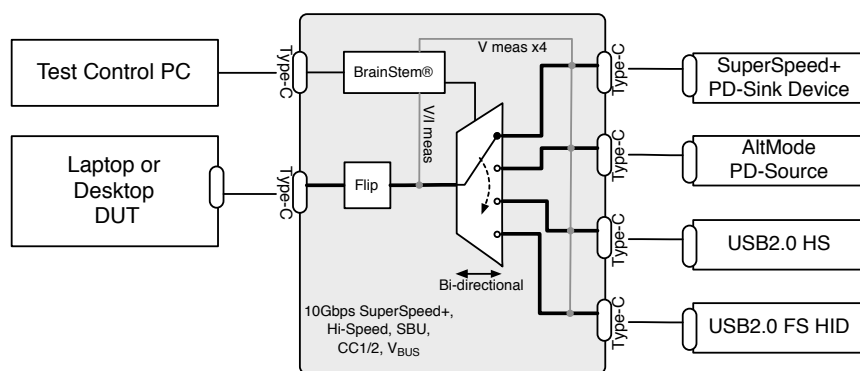
Application Diagrams

Figure 4: Typical testing application for validation against multiple types of devices.
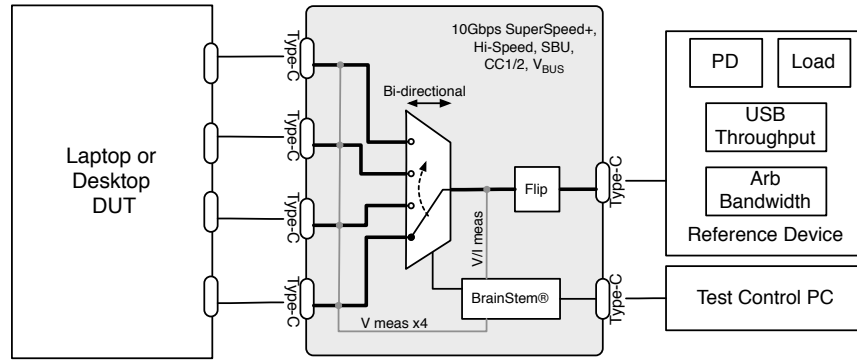
Figure 5: Typical testing application for validation against multiple types of devices.
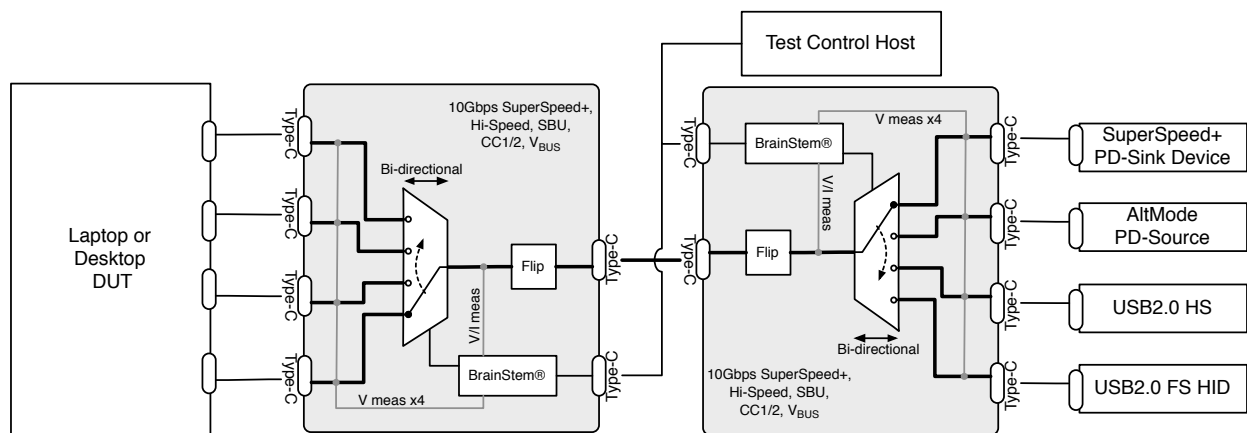
Figure 6: Typical testing application for validating multiple ports against multiple types of devices.[4]

Figure 7: Add Power Delivery charging to a non-PD system using the split mode feature[5]

---

[4] Link budget considerations are crucial for application involving back-to-back USB-C-Switches. Redriver model is recommended.

[5] Split mode can cause irreparable damage to connected devices. Due care should be exercised in setup and application.

Absolute Maximum Ratings[6]

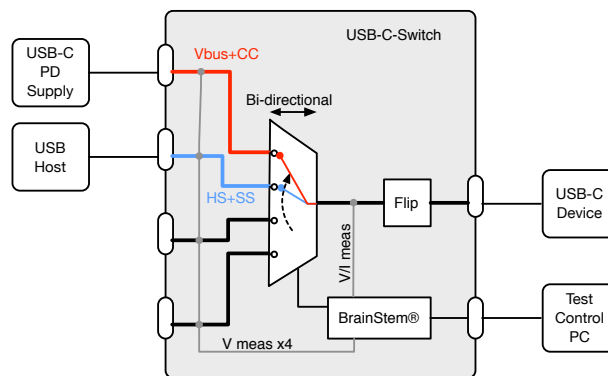Stresses beyond those listed under ABSOLUTE MAXIMUM RATINGS can cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under RECOMMENDED OPERATING CONDITIONS is not implied. Exposure to absolute-maximum rated conditions for extended periods affects device reliability and may permanently damage the device.

Parameter	Minimum	Maximum	Units
Input voltage on V_{BUS} control port pin	-0.3	6.0	V
Voltage on any V_{BUS}, CC pin	-0.3	30	V
V_{BUS} current (bidirectional)	0.0	5.0	A
Voltage on any (SS) data pin	-0.3	2.5	V
Voltage on any USB High Speed (HS) data and SBU pins	-0.3	4.5	V
Altitude	-	2000	M
Overvoltage	-	Overvoltage	-

Table 1: Absolute maximum ratings

Recommended Handling Ratings

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Ambient operating temperature, T_A	Non-Condensing	0	25	50	°C
Relative Humidity Range	Non-Condensing	5	-	95	%RH
Storage temperature, T_{STG}		-10.0	-	85.0	°C
Electrostatic discharge, V_{ESD}	Exceeds IEC 61000-4-2, level 4, air-discharge	-15	-	+15	kV
	Exceeds IEC 61000-4-2, level 4, contact-discharge	-8	-	+8	kV

Table 2: Handling ratings

Recommended Operating Ratings

Specifications are valid at 25°C unless otherwise noted. Indoor use only.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Ambient operating temperature, T_A	Non-Condensing	0	25	50	°C
Relative Humidity Range	Non-Condensing	5	-	95	%RH
Input voltage on V_{BUS} control port pin		4.5	5.0	6.0	V
Voltage on any V_{BUS} pin		0.0	-	20.0	V
V_{BUS} current	Bidirectional	0.0	-	5.0	A
Voltage on SS data pin	Common mode	0.0	-	2	V
	Differential	0.0	-	1.8	V_{pp}
Voltage on any HS data pin		0.0	-	4.3	V
Voltage on any SBU pin		0.0	-	4.3	V
Voltage on any CC pin		0.0	-	5.0	V

Table 3: Recommended operating ratings

---

[6] If the equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

Typical Performance Characteristics

Specifications are valid at 25°C unless otherwise noted. Indoor use only. Sample rates are typically limited by the USB throughput of the host operating system except where bulk capture is supported.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
V$_{BUS}$ common to mux port ON		200	250	350	mΩ
V$_{BUS}$ current measurement resolution		-	1.95	-	mA
V$_{BUS}$ current measurement accuracy		-	±0.5	-	%FS
V$_{BUS}$ voltage measurement resolution		-	8	-	mV
V$_{BUS}$ voltage measurement accuracy		-	±0.2	-	%FS
CCx current measurement resolution		-	976	-	µA
CCx current measurement accuracy		-	±0.5	-	%FS
CCx voltage measurement resolution		-	4	-	mV
CCx voltage measurement accuracy		-	±0.5	-	%FS
Keep-alive charge (KAC) voltage	Sourced from control port V$_{BUS}$	4.5	5.0	5.5	V
Keep-alive charge (KAC) current limit	Constant current mode short circuit to ground	600	800	1000	mA
A5 common to A5 mux DCR		1.0	-	1.1	Ω
B5 common to B5 mux DCR		1.0	-	1.1	Ω
A5 common to B5 mux DCR	Software flip condition	1.0	-	1.1	Ω
B5 common to A5 mux DCR	Software flip condition	1.0	-	1.1	Ω

Table 4: Typical performance characteristics for both models

Passive Model Typical Performance Characteristics

Values presented apply to the full operating temperature range.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
SS data single-ended insertion loss *(Passive model)*	ƒ = 1.6 GHz	-	-4.5	-	dB
	ƒ = 2.5 GHz	-	-8.0	-	dB
	ƒ = 5.0 GHz	-	-11	-	dB
SS data differential OFF isolation	ƒ = 0.3 MHz	-	-40	-	dB
	ƒ = 2.5 GHz	-	-35	-	dB
	ƒ = 5.0 GHz	-	-28	-	dB
SS data channel crosstalk	ƒ = 0.3 MHz	-	-40	-	dB
	ƒ = 2.5 GHz	-	-35	-	dB
	ƒ = 5.0 GHz	-	-28	-	dB
SS data propagation delay		-	3.0	-	ns
SS data intra-pair skew		-	10	-	ps
SS data inter-pair skew		-	30	-	ps
HS data ON resistance		-	9.0	-	Ω
HS data ON resistance imbalance		-	0.5	-	Ω
HS data ON resistance flatness	V=0.0-1.0, VI=30mA	-	1.5	-	Ω
HS data propagation delay		-	0.6	-	ns
HS data OFF isolation		-	-48	-	dB
HS data crosstalk		-	-30	-	dB

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
HS data 3dB bandwidth		-	1200	-	MHz
SuperSpeed SuperSpeed+ data rate	Cable link budget dependent	5	-	10	Gbps
HS data rate	SS data disabled	1	-	480	Mbps

Table 5: Typical performance characteristics for passive model

Redriver Model Typical Performance Characteristics

Specifications are valid at 25°C unless otherwise noted. Indoor use only. Sample rates are typically limited by the USB throughput of the host operating system except where bulk capture is supported. The redriver model is comprised of a daughter-card installed in the passive model. As such, the passive model specifications act as the baseline for the redriver model.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
SuperSpeed receiver equalization gain	5GHz, 1100mV transmitter bias, common to mux ports	-4.4	-	8.6	dB
	5GHz, 1100mV transmitter bias, mux to common ports	-2.1	-	10.5	dB
SuperSpeed input signal detect assert level	10 Gbps PRBS7 pattern, differential	-	80	-	mV$_{pp}$
SuperSpeed input signal detect de-assert level	10 Gbps PRBS7 pattern, differential	-	60	-	mV$_{pp}$
SS low-frequency periodic signaling (LFPS) detect threshold		100	-	300	mV
SS receiver dynamic range	Differential to any SS receiver	-	2000	-	mV$_{pp}$
SS transmitter dynamic range	Differential to any SS transmitter	-	1500	-	mV$_{pp}$
SS output jitter	PRBS7, 10Gbps	-	0.15	-	UI$_{pp}$
High Speed (HS) gain	Receiver EQ Level 0 or Transmitter 0mV	-	0	-	dB
	Receiver EQ Level 1, Transmitter 60mV	-	3	-	dB
	Receiver EQ Level 2, Transmitter 60mV	-	5	-	dB
	Receiver EQ Level 2, Transmitter 80mV	-	9	-	dB

Table 6: Typical performance characteristics for redriver model

Typical Measurements

Passive Model Measurements
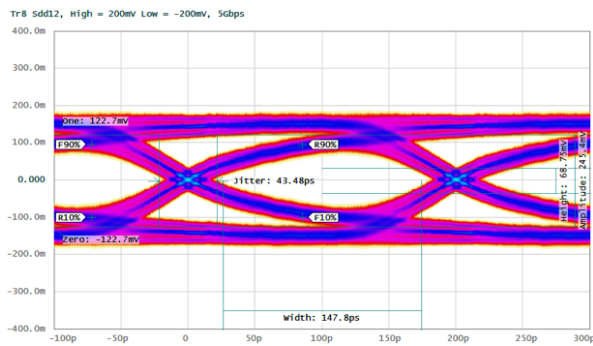

Figure 8: Typical SS data differential insertion loss

Redriver Model Measurements
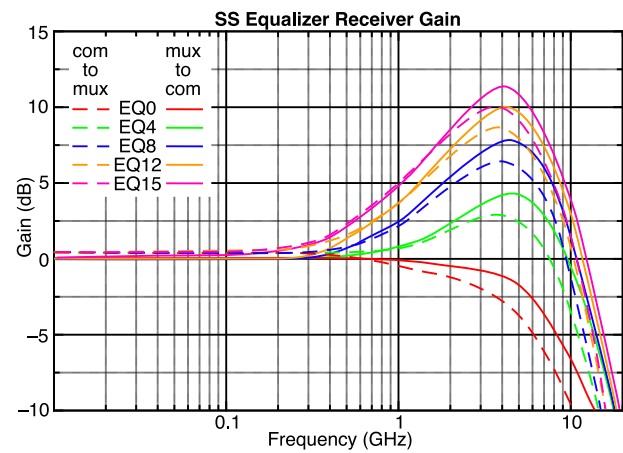
Figure 10: Redriver model SS equalizer receiver gain

Figure 9: Simulated SS data 5Gbps eye diagram

Overview

The USB-C-Switch is a platform to simplify switching of multiple USB Type-C ports. The switch is a bidirectional four-to-one or one-to-four multiplexer (mux) which can create a dedicated connection between a device on the common port and a device on one of the available mux channels. By not using a USB hub, the USB-C-Switch can be truly bidirectional meaning a USB host and device can be attached to either the common or mux side, and that device roles can be swapped. USB-C-Switch is compatible with USB Type-C® (USB-C) applications including link rates up to USB 3.2 Gen 2x1 SuperSpeed+ (10Gbps) and alternate modes (alt-modes), CarPlay® and Android Auto®. Supported alt-modes include HDMI, DisplayPort and digital audio.

At its core the passive model of the switch is an analog mux for USB Hi-Speed (HS), SuperSpeed+ (SS) and side band use (SBU) signals. V_{BUS} and CC signals pass through current and voltage measurement blocks for use in testing and debugging of USB-C systems. The CC lines have USB compliant cable orientation detection circuitry, which enables the USB-C-Switch to properly route signals when using two standard-compliant USB cables. Further, when used with an Acroname Universal Orientation Cable (UOC, part number C38-USBC-UOC), the USB-C-Switch includes circuitry to emulate a cable flip which reverses the apparent cable orientation to connected devices. This programmatic flip feature can be used to automate testing of both cable orientations without manual unplugging, changing orientation and re-inserting a USB-C cable.

The switch is powered and controlled by the USB-C control port, and will appears as a standard USB device when connect to control host. Using the BrainStem software APIs, all features of the USB-C-Switch can be programmatically controlled including selecting a mux channel to be connected to the common port.

Cable Flip

A key feature of the USB-C connector is its symmetric design allowing for insertion in either orientation. This makes the USB-C connector user-friendly yet complicates the development of devices using the USB-C standard. The orientation is defined by the cable or downstream device in the system; more specifically, by components inside of the USB-C male plug of a connection. The USB-C specification makes determining connector orientation a responsibility of the active devices in the system.

Figure 11 shows example block diagrams of the flip feature when connecting a host through a full-featured, non-marked cable to a direct-connected downstream device. Related USB SS, HS and SBU lines are also routed appropriately, though omitted from the diagram for clarity.
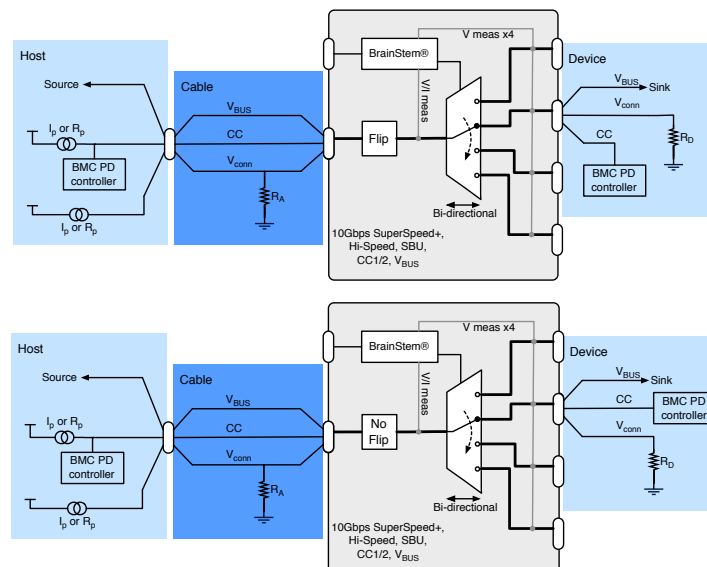
Figure 11: Flip and no-flip setting for full-featured cable and device

With an Acroname UOC cable, the USB-C-Switch enables the unique ability to affect a cable orientation flip. When this orientation flip occurs, it will appear connected devices that the orientation of their connection has reversed. Most USB-C devices with a female socket will include at least one set of muxes in order to route signal to the correct side of the socket based on the

orientation of the cable. These muxes are vital to the end-user orientation agnostic experience of USB-C. When testing such a system it is import to test both orientations to ensure that these internal muxes are functioning. Normally this is done by manually flipping a cable connection, which is time consuming, subjective and error prone. The USB-C-Switch allows flipping of USB-C cable connections to be programmatically automated.

When making connections between devices, as a general rule, ensure that there is only one standard-compliant cable in the connection path between the USB host and USB device. That is, a UOC should be used on either the common port or mux port to enable automated cable flips. The UOC should be connected to the device under test.

When not using the cable flip feature, any standard USB-C cable can be used on both sides of the USB-C-Switch. The orientation of the cables need to be matched in order to facilitate a connection through the switch.

Keep-Alive Charging (KAC)

It is common to use battery powered devices on either side of the USB-C-Switch. When these devices are not in the active path, either on the common or mux side, the device battery may discharge. The USB-C-Switch has the unique feature of Keep-Alive Charging (KAC) for the mux channel connections.

When KAC is enabled, the KAC circuit connects power from the control port V_{BUS} to all non-selected mux channel V_{BUS} lines. KAC power is applied only to inactive mux channels and is not applied to the actively selected mux channel since the actively selected channel has a power path to the common port. KAC is automatically disabled when mux split mode is enabled.

The KAC circuit does not provide any USB power-delivery (USB-PD), USB battery charge specification (BC1.2) or QuickCharge® to these non-selected mux ports. Mux channels with KAC enabled are configured in a dedicated charge port mode (DCP) which is compatible with the CLA protocol from USB v1.1 so most mobile devices will support some level of charging from KAC. KAC is current limited and should a connected device draw more than the allowed current, the KAC circuit will go into a constant current mode, dropping the voltage to maintain the current. The KAC circuit is thermally protected and will disable KAC power outputs if needed. KAC must disabled and re-enabled to restore charging after being disabled on any port due to over-current or over-temperature,.
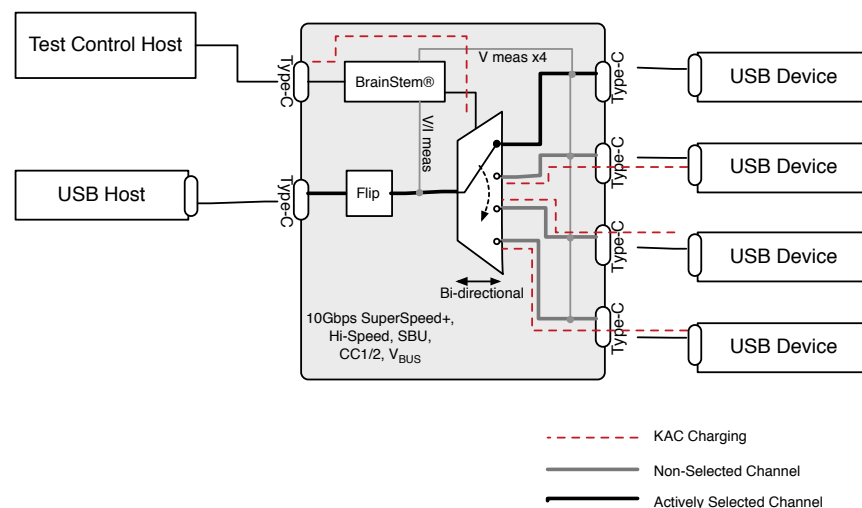

Figure 12: Typical example of KAC charging

Mux Split Mode

The default behavior of the USB-C-Switch is to act as a port selector, where all USB-C lines are connected between the common port and one selected mux channel. In some cases, it is desirable to split the connections in a USB-C cable and route them to different mux paths. A common application is to be able connect a USB device to a host machine for USB data while connecting V_{BUS} charging from a device specific charger.

Split mode gives control over individual signal groups, allowing each group to be connect to a mux channel. V_{BUS} can be connected to any combination of mux channels or disabled on the mux channels. Signal groups under Split control assignment are: V_{BUS}, SSA (TX1+/-, RX1+/-), SSB (TX2+/-, RX2+/-), HSA (D+/-, Side A), HSB (D+/-, Side B), CC1, CC2, SBU1, and SBU2.

When split mode is enabled, V_{BUS} is given a multi-point split capability such that it can be assigned to multiple mux channels concurrently, which is useful for powering multiple devices. Acroname recommends that V_{BUS} be assigned to only one mux channel. Caution should be used with multi-point V_{BUS} assignments as it is possible to apply a V_{BUS} voltage to a device that has not negotiated for high V_{BUS} voltages which could damage connected devices.

When split mode is enabled, USB-C-Switch will automatically disable the Keep-Alive-Charging (KAC) feature.
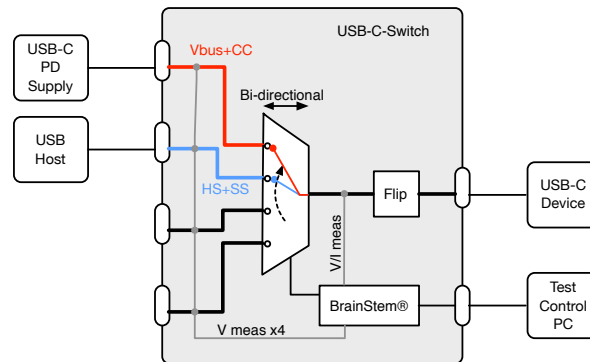

Figure 13: Adding USB-C PD charging capability to a legacy USB host output

CAUTION: Split mode can create connections and configurations not possible or compliant with standard USB equipment. Using this feature could cause unexpected voltages to be applied to devices which may damage connected equipment

Device Drivers

The USB-C-Switch leverages operating system user space interfaces that do not require custom drivers for operation on all modern operating systems including Windows, Linux and MacOS X. With a connection between a host PC and the USB-C control port, the host PC will recognize a USB full-speed device named "USBCSwitch".

Legacy operating systems like Windows 7 may require the installation of a BrainStem USB driver. Installation details on installing USB drivers can be found within the BrainStem Development Kit under the "drivers" folder.

Capabilities and Interfaces

The USB-C-Switch is built on Acroname's BrainStem platform which provides simple, high-level APIs, a real-time embedded runtime engine, and modular expandability. Details of the API functionality unique to the USB-C-Switch are described in the following sections. Refer to BrainStem Reference documentation[7] for generic information about the APIs. See Table 18 for a complete list of supported BrainStem API calls. All shortened code snippets are loosely based on the C++ method calls and are meant to be used as example pseudocode. Reflex methods are not currently supported by USB-C-Switch.

At the highest level, BrainStem devices present a unified device class. The USB-C-Switch uses the USBCSwitch class. Within this device class are many entities, sometimes in arrays of entities, which behave like software classes. Each entity has one or more options (similar to methods) which may take parameters. Most entity options are designed as "getters" and "setters".
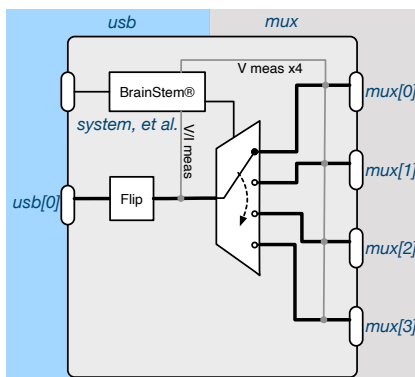

Figure 14: High-level device class division

The USBCSwitch device class logically divides the hardware between the common port and the mux ports as shown in Figure 14. The *usb* entity controls and interfaces to aspects of the common while the *mux* entities controls and interfaces to aspects of the mux ports. Since the USB-C-Switch generally connects the common port to one mux port, the *usb* entity controls many low details of the connection while the *mux* entity generally selects which mux port is enabled. The details of this concept are made clear in each of the entity descriptions.

System Entities

Every BrainStem module includes a single System Entity. The System Entity allows access to configuration settings such as the module address, input voltage, control over the user LED and many more. Please see the Brainstem Reference materials on the website for a full description.

Serial Number

Every USB-C-Switch is assigned a unique serial number at the factory. This facilitates an arbitrary number of USB-C-Switch devices attached to a host computer. The following method call can retrieve the unique serial number for the currently connected device. The BrainStem C++ and python libraries both provide API calls for discovering attached BrainStem devices to facilitate connecting when multiple BrainStem devices are available. This serial number is also presented in the USB device descriptor when connected to host via the control port.

```
stem.system.getSerialNumber(serialNumber)
```

Saving USB Entity Settings

Some entities can be configured and saved to non-volatile memory. This allows a user to modify the startup and operational behavior for the USB-C-Switch away from the factory default settings. Saving system settings preserves the settings to become the new default. Most changes to system settings require a save and reboot before taking effect. USB Boost settings, for example, will not take effect unless a system save operation is completed, followed by a reset or power cycle. Pressing the reset button will return all settings to factory defaults. Use the following command to save changes to system settings before reboot:

```
stem.system.save()
```

---

[7] BrainStem API reference https://acroname.com/reference/

Saved Configurations	
USB Mode (*usb*)	Mux Configuration (*mux*)
Mux Split Mode (*mux*)	Equalizer Configuration (*equalizer*)

Table 7: Saved entities

USB Entity

The *usb* entity provides a mechanism to control and monitor all USB functionality on the common port. Individual parts of the USB connection can be manipulated through the *usb* entity. For example, enable/disable USB data and V_{BUS} lines, measure current and voltage on V_{BUS}, V_{CONN}, and CC. The USB-C-Switch has one *usb* entity class. It uses the mux entity to select one of the 4 mux channels to which to connect the enabled USB signals.

The *usb* entity splits the USB connection into tree going from most generic to most specific with *usb* entity options at each level. Higher levels of the tree can be used to cause simultaneous changes on the lower levels. The tree structure is port(VBUS, data(HS, SS), USB-C(CC1, CC2, SBU)).

The *usb.setPortEnable/Disable* entity option allows for manipulating all parts of the USB connection (HS data, SS data, both CC and SBU lines, and V_{BUS} lines) simultaneously.

```
stem.usb.setPortEnable(channel)
stem.usb.setPortDisable(channel)
```

Where channel is always 0 for the USB-C-Switch. Further examples of the *usb* entity will always show the channel option as 0.

Manipulating USB data lines (HS and SS) simultaneously is done by calling:

```
stem.usb.setDataEnable(0)
stem.usb.setDataDisable(0)
```

Manipulating the HS or SS data lines is done by calling:

```
stem.usb.setHiSpeedDataEnable(0)
stem.usb.setHiSpeedDataDisable(0)
stem.usb.setSuperSpeedDataEnable(0)
stem.usb.setSuperSpeedDataDisable(0)
```

Manipulating the V_{BUS} line is done by calling:

```
stem.usb.setPowerEnable(0)
stem.usb.setPowerDisable(0)
```

The automatic orientation detection and connection functionality is interface with:

```
stem.usb.setConnectMode(0, mode)
stem.usb.getConnectMode(0, mode)
```

where *mode* is a Boolean value of 0 or 1.

Manipulating the CC lines is done by calling:

```
Stem.usb.setCC1Enable(0, enabled)
stem.usb.setCC2Enable(0, enabled)
stem.usb.getCC1Enable(0, enabled)
stem.usb.getCC1Enable(0, enable)
```

where *enable* is a Boolean value of 0 or 1.

CC line current and voltage can be measured with:

```
stem.usb.getCC1Voltage(0, µV)
stem.usb.getCC2Voltage(0, µV)
stem.usb.getCC1Current(0, µA)
stem.usb.getCC2Current(0, µA)
```

where positive current is power transfer from the common port to the mux port.

Manipulating the SBU lines is done by calling:

```
stem.usb.setSBUEnable(0, enabled)
stem.usb.getSBUEnable(0, enabled)
```

where enable is a Boolean value of 0 or 1.

V_{BUS} voltage and current through the switch's V_{BUS} lines can be measured with:

```
stem.usb.getPortVoltage(0,µV)
stem.usb.getPortCurrent(0,µA)
```

where positive current is power transfer from the common port to the mux port.

Cable Flip

The USB-C-Switch can simulate a cable flip by electrically switching the CC/V_{CONN} and SBU lines between side-A and side-B of the USB-C female sockets. USB data lines are also swapped accordingly. This flip can be done with:

```
stem.usb.getCableFlip(setting)
stem.usb.setCableFlip(setting)
```

where *setting* parameter is an integer value of 0 or 1, where 0 is normal and 1 is flipped.

Individual functional groups of the USB connection can be flipped by using the *portMode* option.

USB Port Mode

The *portMode* option provides a bitmapped setting for granular control of the individual connections. The *portMode* option is the desired mode of the port. The companion option of *portState* is used to interface with the actual state of the port.

```
Stem.usb.getPortMode(0, mode)
stem.usb.setPortMode(0, mode)
```

where *mode* is 32-bit word, defined in Table 8 where 0 isdisabled and 1 is enabled.

Bit	Port Mode Bit Map
0	Reserved
1	Reserved
2	Keep Alive Charging Enable
3	Reserved
4	HS Side A Data enable
5	HS Side B Data enable
6	V$_{BUS}$ enable
7	SS Lane 1 Data enable
8	SS Lane 2 Data enable
9:10	Reserved
11	Auto Connect enable
12	CC1 enable
13	CC2 enable
14	SBU enable
15	CC Flip enable
16	Super-Speed Flip enable
17	SBU Flip enable
18	Hi-Speed Flip enable
19	CC1 Current Injection enable
20	CC2 Current Injection enable
21:31	Reserved

Table 8: Port mode bit map

USB Port Operational State

The *portState* option provide an interface to the state of the common port and internals of the USB-C-Switch system.

```
stem.usb.getPortState(0, state)
```

where *mode* is 32-bit word, defined in Table 9 where 0 is disabled and 1 is enabled.

Bit	Port State Bit Map
0	V$_{BUS}$ enable
1	HS Side A Data enable
2	HS Side B Data enable
3	SBU enable
4	SS Lane 1 Data enable
5	SS Lane 2 Data enable
6	CC1 enable
7	CC2 enable
8:9	Common port orientation status

Bit	Port State Bit Map
10:11	Mux channel orientation status
12:13	Reserved
14	CC Flip enable
15	Super-Speed Flip enable
16	SBU Flip enable
17	Reserved
19:18	Daughter-Card status
22:20	Error Flag
23	Connection Established
24:25	Reserved
26	CC1 Current Injection
27	CC2 Current Injection
28	CC1 Pulse detect
29	CC2 Pulse detect
30	CC1 Logic state
31	CC2 Logic state

Table 9: Port state bit map

USB Alt Mode Configuration

The redriver model USB-C-Switch provides an intermediary receiver and amplifier on the HS and SS data lines. Various alt-modes such as DisplayPort require different directional uses of the SS data lines. As such, it is required to define the alt-mode and direction of the connection. These modes are responsible for setting the direction of the SS data lines and related SBU lines.

```
stem.usb.getAltModeConfig(0, configuration)
```

```
stem.usb.setAltModeConfig(0, configuration)
```

where *configuration* is an integer value defined in Table 10. Details of the pin mapping and data direction for each configuration is shown in Table 19.

Index	Alt Mode Configuration
0	USB 3.1 Disabled
1	USB 3.1 Enabled
2	4 Lane DisplayPort Host on Common Port
3	4 Lane DisplayPort Host on Mux Port
4	2 Lane DisplayPort with USB 3.1 – Host on Common Port
5	2 Lane DisplayPort with USB 3.1 – Host on Mux Port
6	2 Lane DisplayPort Host on Common Port with USB 3.1 Inverted

Index	Alt Mode Configuration
7	2 Lane DisplayPort Host on Mux Port with USB 3.1 Inverted

Table 10: Alt-mode configurations

Bit	Mux Split Mode Bit Map
23	V$_{BUS}$ enable CH3
24:31	Reserved

Table 11: Mux Split Mode Result Bitwise Description

Mux Entity

The *mux* entity primarily selects one active mux port to connect to the common port using the *channel* option:

```
stem.mux.setChannel(channel)

stem.mux.getChannel(channel)
```

where *channel* is an index 0-3.

Mux Configuration

Default configuration of the mux is to switch all enabled USB-C lines to a single mux channel. If desired, the switch can split the USB-C functional groups and route them to selected mux ports. This feature is referred to as "split mode". Default or split modes can be enabled with:

```
stem.mux.getConfiguration(config)
stem.mux.setConfiguration(config)
```

where *config* is 0 for default and 1 for split mode.

Mux Split Mode

After enabling split mode, USB-C functional groups can be individually assigned to separate mux channels with:

```
stem.mux.getSplitMode(splitMode)
stem.mux.setSplitMode(splitMode)
```

where *splitMode* is 32-bit word, defined in Table 11. Each bit pair is a 2-bit binary number from 0-3 representing the mux port to which to route the functional signal group. V$_{BUS}$ uses 4 bits to define which mux ports are connected to the common port V$_{BUS}$ lines.

Bit	Mux Split Mode Bit Map
0:1	SBU1
2:3	SBU2
4:5	CC1
6:7	Reserved
8:9	CC2
10:11	Reserved
12:13	HS Side A Data
14:15	HS Side B Data
16:17	SS Lane 1 Data
18:19	SS Lane 2 Data
20	V$_{BUS}$ enable CH0
21	V$_{BUS}$ enable CH1
22	V$_{BUS}$ enable CH2

Equalizer Entity

The redriver model of the switch provides two equalizer entities. They provide programmatic control over linear equalizers and amplifiers (aka: redrivers) connected to the HS and SS data lines. These equalizer entities split the configuration between receiver-side and transmitter-side settings allowing for compensation of signal integrity loss due to cable quality, length and insertion losses. However, some of the settings can have combined effects between receiver and transmitter modes. The two *equalizer* entities are indexed to their respective data lines as defined in Table 12:

Index	Equalizer Entity Mapping
0	USB2 High Speed
1	USB3 SuperSpeed

Table 12: Equalizer entity index mapping

The transmitter is responsible for driving and selectively amplifying the signals traveling out redriver hardware after any receiver-side equalization. Each equalizer entity has transmitter options of:

```
stem.equalizer[x].setTransmitterConfig(config)
stem.equalizer[x].getTransmitterConfig(config)
```

The receiver attempts to compensate for distortion of the incoming signal. Each equalizer entity has receiver options of:

```
stem.equalizer[x].setReceiverConfig(chan, config)
stem.equalizer[x].getReceiverConfig(chan, config)
```

where the *chan* parameter options are defined in Table 13.

Value	Receiver Channel
0	Applies setting to both common and mux
1	Applies settings to mux side
2	Applies settings to common side

Table 13: Receiver channels

High Speed Redriver Configuration

Due to the half-duplex nature of USB2 data lines there is only one receiver and transmitter setting for both the common and mux ports. In addition, since the transmitter and receiver are tightly coupled, the linear gain achieved by transmitter setting varies with the equalizer receiver configuration. Approximate gains for example configurations are shown in the specifications table.

HS Equalizer Transmitter Configuration

The HS equalizer entity transmitter option controls the gain applied to HS signals only; USB Low Speed (LS) and Full Speed (FS) signals are unaffected and uncompensated. This option changes the DC boost applied to HS signals which can help achieve sharper rising edges. The allowed values are shown in Table 14. If the HS equalizer transmitter option is set to 0mV the HS redriver is disabled regardless of the HS equalizer receiver configuration.

```
stem.equalizer[0].setTransmitterConfig(config)
stem.equalizer[0].getTransmitterConfig(config)
```

Value	High Speed Transmitter Configuration
0	40mV DC Boost
1	60mV DC Boost
2	80mV DC Boost
3	0mV DC Boost (disabled)

Table 14: High Speed Transmitter configurations

HS Equalizer Receiver Configuration

The *chan* parameter of the HS *equalizer receiver* option can only be 0 because the HS data lines are half-duplex. All other values will result in an error return.

The HS *equalizer receiver* option configurations control the sensitivity of the redriver to incoming HS signals. The effect of this change in sensitivity can be considered a variable AC boost tuned to the specific HS signal applied. Table 15 shows the available options. Setting the HS equalizer receiver option to Level 0 will disable the HS redriver regardless of the HS entity transmitter option configuration.

```
stem.equalizer[0].setReceiverConfig(0, config)
stem.equalizer[0].getReceiverConfig(0, config)
```

Value	High Speed Receiver Equalization
0	Level 1
1	Level 2

2	Level 0 (disabled)

Table 15: High Speed receiver configurations

SuperSpeed Redriver Configuration

SS Equalizer Transmitter Configuration

The SS *equalizer transmitter* option controls various transmitter gains for each side of the full-duplex SS data lines. That is, each configuration combines the transmitter gain and approximate peak-to-peak voltage for both the common and mux side transmitters. The available options are show in Table 16.

Value	Mux Side	Com Side	Range
0	+1db	+0db	$900mV_{pp}$
1	+0db	+1db,	$900mV_{pp}$
2	+1db	+1db	$900mV_{pp}$
3	+0db	+0db	$900mV_{pp}$
4	+0db	+0db	$1100mV_{pp}$
5	+1db	+0db	$1100mV_{pp}$
6	+0db	+1db	$1100mV_{pp}$
7	+2db	+2db	$1100mV_{pp}$
8	+0db	+0db	$1300mV_{pp}$

Table 16: SuperSpeed Transmitter Configurations

SS Equalizer Receiver Configuration

The SS *equalizer receiver* option controls the receiver gain. The actual receiver gain is dependent on the alt-mode configuration and the port data direction (mux to common vs common to mux). There are independent receiver gain settings for the common and mux ports of the switch. Gains across settings, direction, and frequency is shown in Figure 10.

Value	SS Receive Gain Level
0 - 15	Increasing levels of gain

Table 17: SuperSpeed redriver receiver configurations

USB-C-Switch Supported Entity Methods Summary

Detailed entity class descriptions can be found in the BrainStem Reference (https://acroname.com/reference/entities/index.html). A summary of USB-C-Switch class options are shown below. Note that when using Entity classes with a single index (aka, 0), the index parameter can be dropped. For example:

```
stem.system[0].setLED(1) → stem.system.setLED(1)
```

Entity Class	Entity Option	Variable(s) Notes
store[0-1]	getSlotState	
	loadSlot	

	unloadSlot	
	slotEnable	
	slotDisable	
	slotCapacity	
	slotSize	
system[0]	save	
	reset	
	setLED	
	getLED	
	getInputVoltage	
	getVersion	
	getModuleBaseAddress	
	setHBInterval	
	getHBInterval	
	getModule	
	getSerialNumber	
	getRouter	
	getModel	
usb[0]	setPortEnable	
	setPortDisable	
	setDataEnable	
	setDataDisable	
	setHiSpeedDataEnable	
	setHiSpeedDataDisable	
	setSuperSpeedDataEnable	
	setSuperSpeedDataDisable	
	setPowerEnable	
	setPowerDisable	
	getPortVoltage	
	getPortCurrent	
	getPortMode	
	setPortMode	
	getPortState	
	setCableFlip	
	getCableFlip	
	setConnectMode	
	getConnectMode	
	setCC1Enable	
	getCC1Enable	
	setCC2Enable	
	getCC2Enable	
	getCC1Voltage	
	getCC2Voltage	

	getCC1Current	
	getCC2Current	
	setSBUEnable	
	getSBUEnable	
mux[0]	setEnable	
	getEnable	
	setChannel	
	getChannel	
	getConfiguration	
	setConfiguration	
	getSplitMode	
	setSplitMode	
	getVoltage	Channels 0-3
equalizer[0-1]	setReceiverConfig	
	getReceiverConfig	
	setTransmitterConfig	
	getTransmitterConfig	

Table 18: Supported USB-C-Switch BrainStem Entity API Methods[8]

---

[8] See BrainStem software API reference at https://acroname.com/reference/ for further details about all BrainStem API methods and information.

Pinouts and Connectivity

USB Type-C Connector Overview

The USB-C-Switch uses standard USB pin outs for the Type-C female receptacles shown in Figure 15. The side-A and side-B USB HS D+ and D- are separately passed through the USB-C-Switch. The common port to mux port pin mapping for normal and flip modes is shows in Figure 16.

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
GND	TX1+	TX1-	VBUS	CC1	D+	D-	SBU1	VBUS	RX2-	RX2+	GND

GND	RX1+	RX1-	VBUS	SBU2	D-	D+	CC2	VBUS	TX2-	TX2+	GND
B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1

Receptacle (Front View)

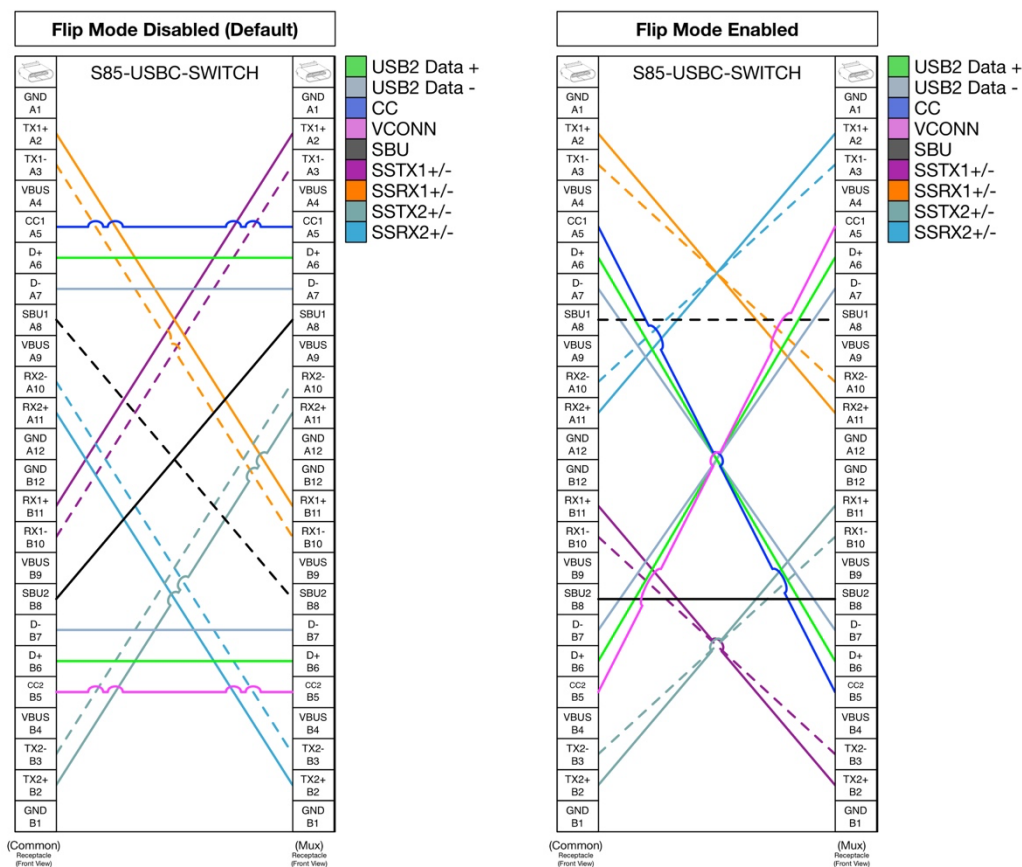*Figure 15: USB type-C receptacle pin out*

Figure 16: Common to mux port pin mappings

Redriver Model Alt-Mode Configurations

For alt-modes, the pin mappings and directions may affect connectivity and functionality. In many cases, the connected devices should simply negotiate through the switch. With some alt-modes, some of the functional groups need to be assigned a specific direction.

Common Port Pin								Mux Port Pin Normal	Mux Port Pin Flipped
Redriver Config	USB 3.1	4 Lane DisplayPort Host on Common	4 Lane DisplayPort Host on Mux	2 Lane DisplayPort Host on Mux with USB3.1	2 Lane DisplayPort Host on Common with USB3.1	2 Lane DisplayPort Host on Common with USB 3.1 Inverted	2 Lane DisplayPort Host on Mux with USB 3.1 Inverted	**Color Key** USB HS / USB SS / DisplayPort (alt-mode)	
A2	←	→	←	←	→	→	←	B11	A11
A3	←	→	←	←	→	→	←	B10	A10
A10	→	→	←	←	→	→	←	B3	A3
A11	→	→	←	←	→	→	←	B2	A2
B2	←	→	←	←	←	→	→	A11	B11
B3	←	→	←	←	←	→	→	A10	B10
B10	→	→	←	→	→	←	←	A3	B3
B11	→	→	←	→	→	←	←	A2	B2
A8	↔	↔	↔	↔	↔	↔	↔	B8	A8
B8	↔	↔	↔	↔	↔	↔	↔	A8	B8

Table 19: Redriver model pin function and direction

Physical Connections and Interface

LED Indicators

On common side of the USB-C-Switch there is a set of indicators that show control information and connectivity status. The meaning and location of each LED are described in the following tables and diagrams.

| LED Name	Color	Description
User	Blue	Can be manipulated through the available APIs
Power/ Heartbeat	Red/Green	Red indicates system is powered. Flashing green is the heartbeat which indicates an active software connection. Pulses at a rate determined by the system heartbeat rate to indicate an active BrainStem link.
Side A USB Status	Green/Yellow	See Figure 12 for status indications.
Side B USB Status	Green/Yellow	
Channel 0 Status	Blue	
Channel 1 Status	Blue	Indicates Mux Channel selection. Disabled when Split mode is enabled.
Channel 2 Status	Blue	
Channel 3 Status	Blue	
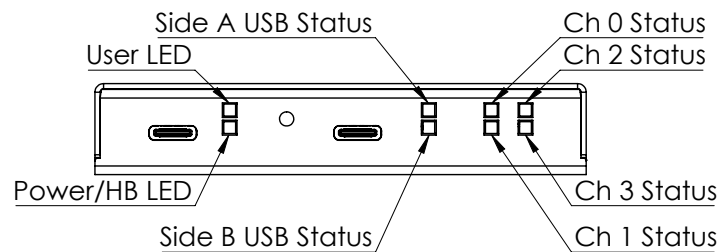
Figure 17: LED positions
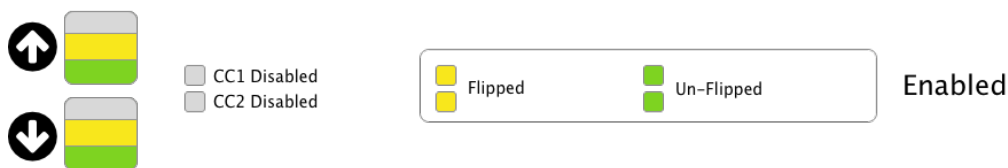
Figure 18: LED status

USB Connections

The rear of the USB-C-Switch has two USB Type-C connections – BrainStem control/power, and the single port side of the switch, referred to as the common port.
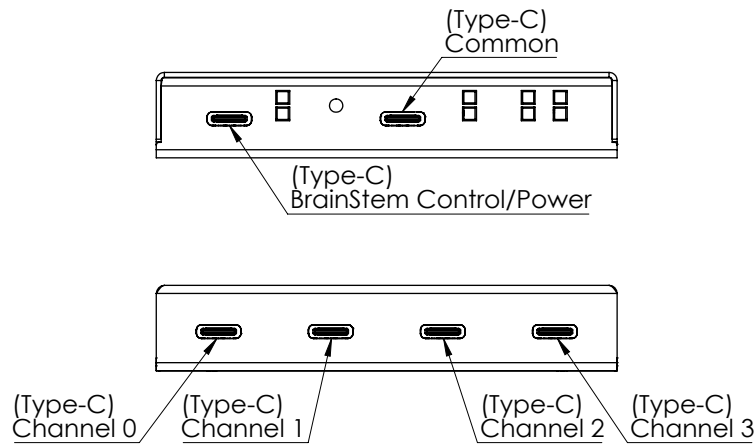
Figure 19: USB-C connector names

Power Input

Power for the USB-C-Switch is provided by the V_{BUS} line on the control port. This port supports USB Power Delivery 1.1 (USB-PD) high current mode of 5V at 3A. See Table 3: Recommended operating ratings for input voltage and power requirements.

Unit Reset

The USB-C-Switch can be reset to factory default settings using the reset button on the control side. Pressing the reset button once will restart the USB-C-Switch as if it had been power cycled. To restore factory default settings, press the reset button two times within 5 seconds.
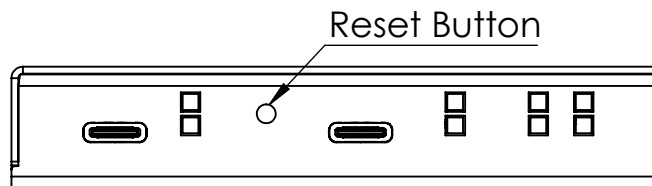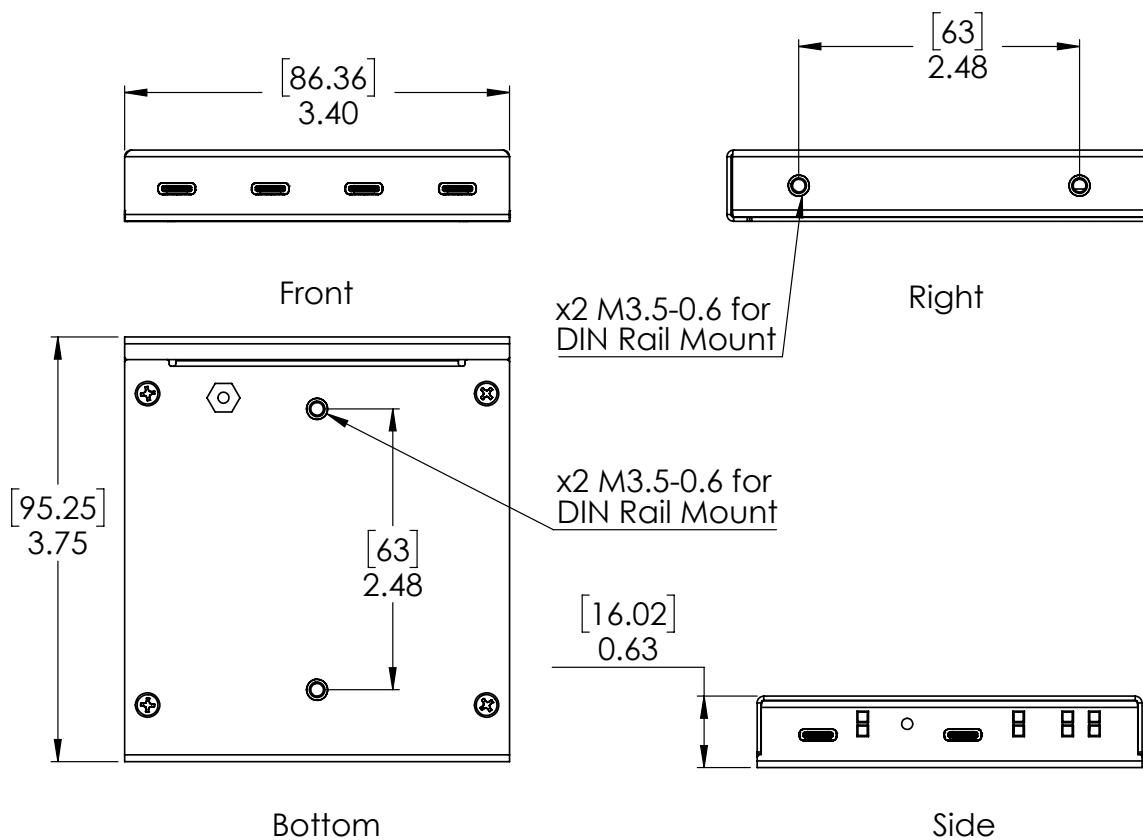

Figure 20: Reset button location

Mechanical

Dimensions are shown in inches [mm]. 3D CAD models available from https://acroname.com.

$$\begin{bmatrix} 86.36 \end{bmatrix}$$
3.40

Front

$$\begin{bmatrix} 63 \end{bmatrix}$$
2.48

Right

x2 M3.5-0.6 for
DIN Rail Mount

$$\begin{bmatrix} 95.25 \end{bmatrix}$$
3.75

x2 M3.5-0.6 for
DIN Rail Mount

$$\begin{bmatrix} 63 \end{bmatrix}$$
2.48

$$\begin{bmatrix} 16.02 \end{bmatrix}$$
0.63

Bottom

Side

DIMENSIONS: IN[MM]
SCALE: 2:3

Figure 21: USB-C-Switch Mechanical Dimensions

DIN Rail Mounting

DIN rail mounting provisions have been designed into the USB-C-Switch case. Holes for a DIN rail clip/adapter are provided to allow mounting of the switch to standard DIN rails. Mounting clip hardware is available separately in a kit from Acroname: part number C31-DINM-1.

The USB-C-Switch can be mounted in two positions as shown in Figure 17.

Warning: Care should be taken to only use clip mounting hardware included by Acroname. Longer screws will cause irreparable damage to the USB-C-Switch.
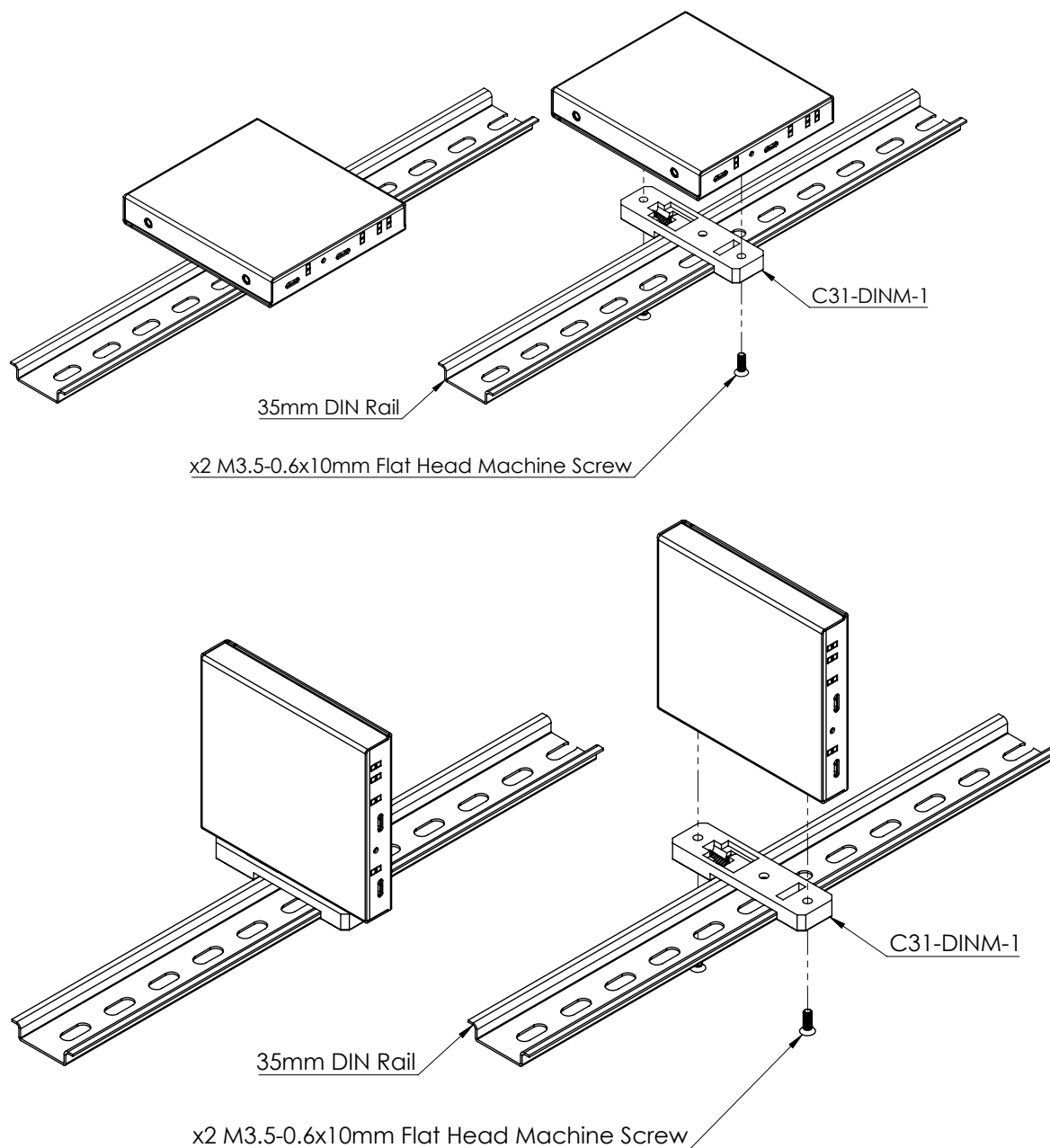

C31-DINM-1

35mm DIN Rail

x2 M3.5-0.6x10mm Flat Head Machine Screw

C31-DINM-1

35mm DIN Rail

x2 M3.5-0.6x10mm Flat Head Machine Screw

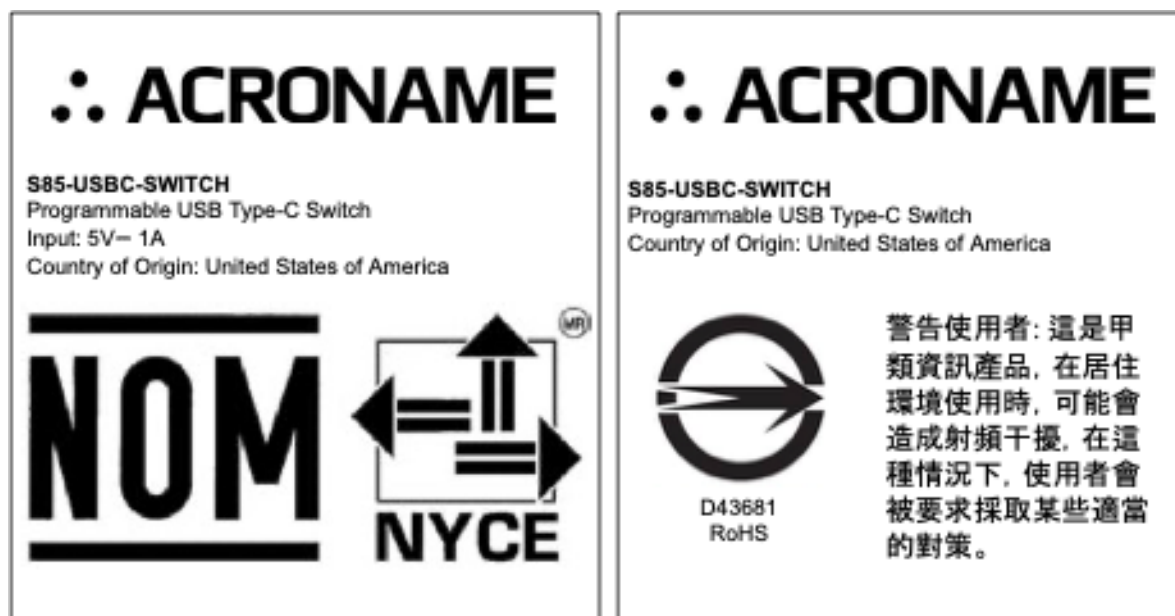*Figure 22: USB-C-Switch DIN Rail mounting*

FCC Compliance Statement

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and receiver.

- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

- Consult the dealer or an experienced radio/TV technician for help.

```
This device complies with part 15 of
FCC Rules.  Operation is subject to the
following two conditions; (1) This
device may not cause harmful
interference, and (2) this device must
accept any interference received,
including interference that may cause
undesired operation.
```

Compliance Information

BSMI RoHS Table

設備名稱： USB-C-Switch ， Equipment name 型號（型式）：S85-USBC-SWITCH Type designation (Type)						
單元Unit	限用物質及其化學符號 Restricted substances and its chemical symbols					
	鉛Lead (Pb)	汞Mercury (Hg)	鎘Cadmium (Cd)	六價鉻 Hexavalent chromium (Cr^{+6})	多溴聯苯 Polybrominated biphenyls (PBB)	多溴二苯醚 Polybrominated diphenyl ethers (PBDE)
機殼 Chassis	○	○	○	○	○	○
電路板 PCB Board	○	○	○	○	○	○
連接器 Connector	○	○	○	○	○	○
電子元件 Electronic Component	○	○	○	○	○	○

備考1.〝超出0.1 wt %〞及〝超出0.01 wt %〞係指限用物質之百分比含量超出百分比含量基準值。

Note 1： "Exceeding 0.1 wt %" and "exceeding 0.01 wt %" indicate that the percentage content of the restricted substance exceeds the reference percentage value of presence condition.

備考2.〝○〞係指該項限用物質之百分比含量未超出百分比含量基準值。

Note 2： "○" indicates that the percentage content of the restricted substance does not exceed the percentage of reference value of presence.

備考3.〝—〞係指該項限用物質為排除項目。

Note 3：The "–" indicates that the restricted substance corresponds to the exemption.

Product Support

Questions about the product operation or specifications are welcome through Acroname's contact portals. Software downloads, reference API and application examples are available online at:

https://acroname.com/support

Direct communication and additional technical support are available at:

https://acroname.com/contact-us

Acroname, Inc
2741 Mapleton Avenue
Boulder, CO, USA 80304-3837
Phone: +1-720-564-0373

Acroname and BrainStem are registered trademarks of Acroname, Inc. All other trademarks are property of their respective owners.

Document Revision History

All major documentation changes will be marked with a dated revision code

Revision	Date	Engineer	Description
0.1	January 2017	JTD	Pre-Release
0.2	July 2017	JLG	Preliminary release
0.3	September 2017	JRS	API updates to preliminary release
1.0	September 2018	LCD	Release and update for hardware, API enhancements
1.1	November 2018	LCD	Corrected support for reflex method
1.2	May 2019	MJK	Added initial documentation support for Redriver
1.3	September 2019	TDH	Corrected typo
1.4	February 2021	MJK	Contact information for technical support.
1.5	June 2021	JLG	Clean up; update loss specifications
1.6	October 2021	MJK	Added altitude, overvoltage information.
1.7	April 2022	FEC	Included NOM label
1.8	April 2022	FEC	Included BMSI Label and BMSI RoHS Table; Changed recommended operating range for VBUS