



## 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 a Common Port to one of 4 available Mux Channel 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 AndroidAuto® testing

## Features

- Selectively connect one USB Type-C connection to any one of 4 ports
- Bi-directional (supports 1:4 or 4:1 configurations)

- All ports support USB 3.1 Gen 2 link speeds up to 10Gbps
- All ports support USB PD profiles up to 100W (20V, 5A)
- Execute USB-C cable flip via software control<sup>1</sup>
- Supports pass through of USB Alt-Modes (DisplayPort, HDMI and Digital Audio)
- HS Data, SS Data,  $CC/V_{CONN}$ , SBU, and  $V_{BUS}$  power can be independently enabled/disabled
- HS Data, SS Data,  $CC/V_{CONN}$ , SBU, and  $V_{BUS}$  power can be independently routed to any mux channel
- Measure  $V_{BUS}$  voltage on each port
- Measure  $V_{CONN}$  voltage and current
- Measure  $V_{BUS}$  current on selected port
- Configurable to be completely passive
- Dedicated control port for software control independent of the switched ports
- DIN-rail mountable
- Certified to withstand  $\pm 15kV$  ESD strikes (IEC61000-4-2 level 4)

## Description

The USB-C-Switch gives engineers advanced control of USB connections in testing and development applications.

The USB-C-Switch hub architecture consists of several layers of internal switches to achieve the 4:1 switch and USB line control functionality. All USB-C signals, including CC and SBU, are passed through the USB-C-Switch. Data link, power negotiations and power between USB devices are provided by the attached devices themselves, allowing the USB-C-Switch to be used bi-directionally in 1:4 or 4:1 configurations.

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

Each USB channel implements independently and separately switched HS, SS,  $V_{BUS}$ , CC and SBU lines. Pin interfaces are protected against reverse polarity and over-voltage. The device and all connections are designed to operate from 0°C to 50°C ambient with no external cooling or fans.

Each USB-C-Switch is uniquely addressable and controllable from a host PC via the selected USB host input. Built with Acroname’s BrainStem® platform, the USB-

<sup>1</sup> Require use of one Acroname Universal Orientation Cable (UOC), C38-USBC-UOC

C-Switch is easily controlled over USB with simple high-level APIs in C, C++, Python and LabView.

## Block Diagram

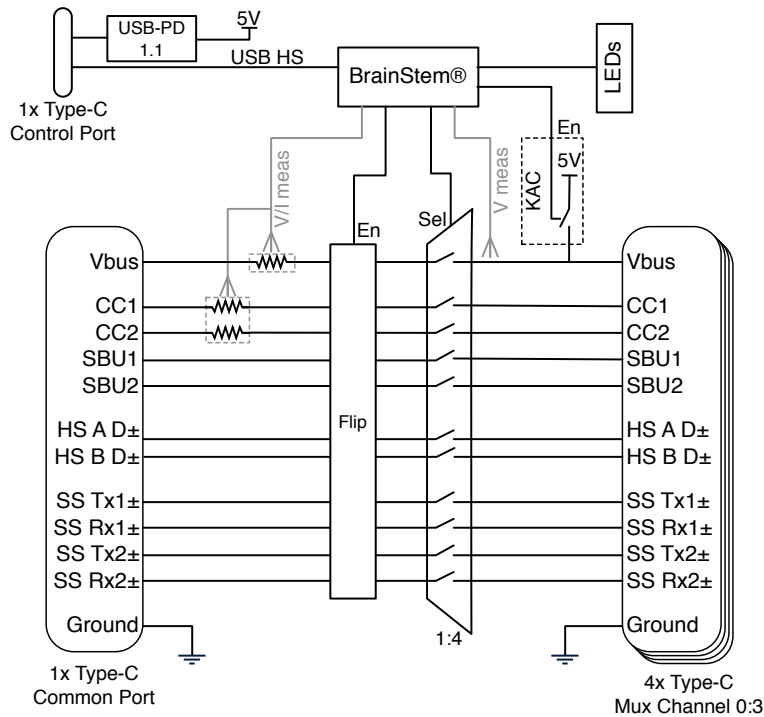


Figure 1: USB-C-Switch Functional Block Diagram

## Application Diagrams

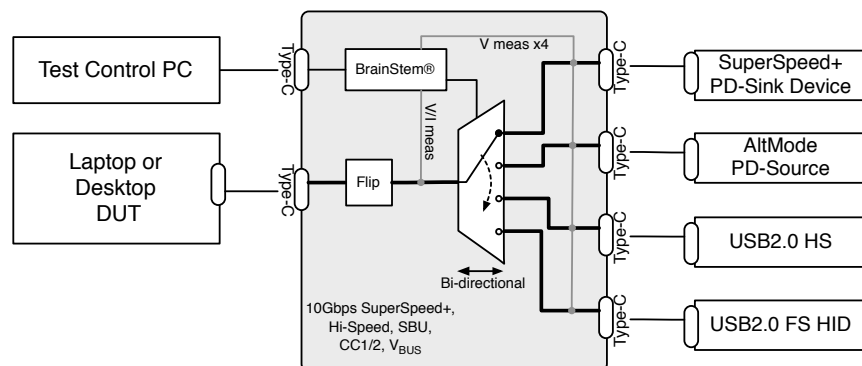


Figure 2: Typical testing application for validation against multiple types of devices.

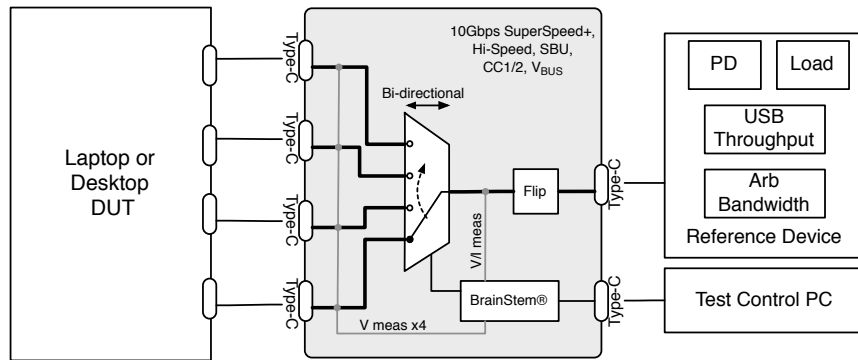


Figure 3: Typical testing application for validation against multiple types of devices.

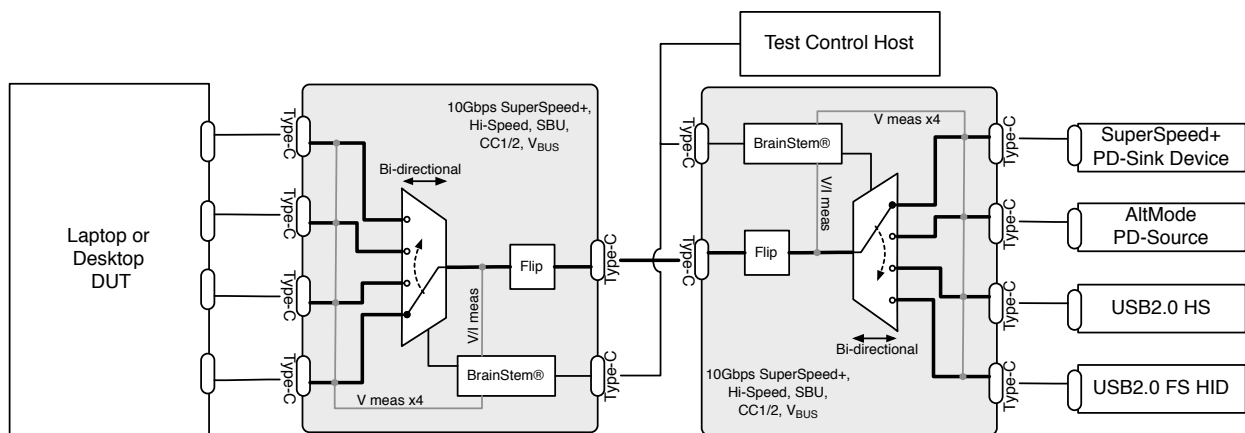


Figure 4: Typical testing application for validating multiple ports against multiple types of devices.

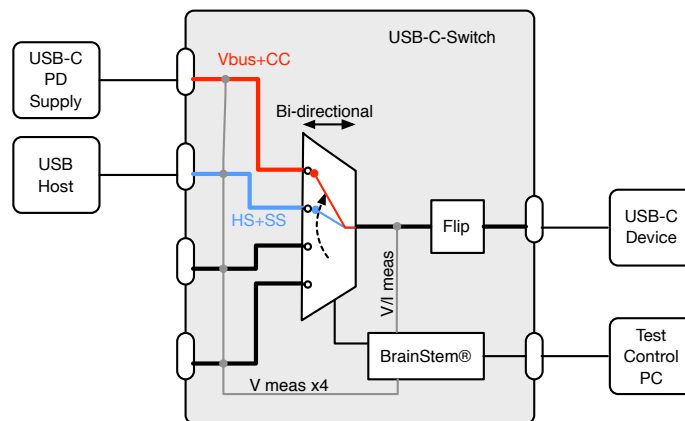


Figure 5: Add Power Delivery charging to a non-PD system using the advanced Split feature

## Absolute Maximum Ratings

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	30.0	V
$V_{BUS}$ current (bi-directional)	0.0	5.0	A
Voltage on any SuperSpeed+ (SS) data pin	-0.3	2.5	V
Voltage on any USB HiSpeed (HS) data and SBU pins	-0.3	4.5	V

Table 1: Absolute Maximum Ratings

## Handling Ratings

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Ambient operating temperature, $T_A$	Non-Condensing	0.0	25.0	50.0	°C
Storage temperature, $T_{STG}$		-10.0	-	85.0	°C
Electrostatic discharge, $V_{ESD}$	Meets IEC 61000-4-2, level 4, air-discharge	-15	-	+15	kV
	Meets IEC 61000-4-2, level 4, contact-discharge	-8	-	+8	kV

Table 2: Handling Ratings

## Recommended Operating Ratings

Values presented apply to the full operating temperature range.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Input voltage on $V_{BUS}$ control port pin		4.0	5.0	6.5	V
Voltage on any $V_{BUS}$ pin		0.0	-	20.0	V
$V_{BUS}$ current	Bi-directional	0.0	-	5.0	A
Voltage on SS data pin	Common mode	0.0	-	2	V
	Differential	0.0	-	1.8	V <sub>pp</sub>
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
Keep-alive charge (KAC) current			500	600	mA

Table 3: Recommended Operating Ratings

## Typical Performance Characteristics

Values presented apply to the full operating temperature range.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
V <sub>BUS</sub> common to mux port ON resistance		200	250	350	mΩ
V <sub>BUS</sub> current measurement resolution		-	1.95	-	mA
V <sub>BUS</sub> current measurement accuracy		-	±0.5	-	%
V <sub>BUS</sub> voltage measurement resolution		-	8	-	mV
V <sub>BUS</sub> voltage measurement accuracy		-	±0.2	-	%
CCx current measurement resolution		-	976	-	μA
CCx current measurement accuracy		-	±0.5	-	%
CCx voltage measurement resolution		-	4	-	mV
CCx voltage measurement accuracy		-	±0.5	-	%
Keep-alive charge (KAC) voltage	Sourced from control port V <sub>BUS</sub>	4.5	5.0	5.5	V
Keep-alive charge (KAC) current limit	Constant current mode short circuit to ground	600	800	1000	mA
SS data differential insertion loss	<i>f</i> = 0.3 GHz preliminary	-	-1.6	-	dB
	<i>f</i> = 1.6 GHz preliminary	-	-3.2	-	dB
	<i>f</i> = 2.5 GHz preliminary	-	-4.2	-	dB
	<i>f</i> = 4 GHz preliminary	-	-5.9	-	dB
	<i>f</i> = 5 GHz preliminary	-	-7.1	-	dB
SS data differential retrun loss	<i>f</i> = 0.3 MHz preliminary	-	-25	-	dB
	<i>f</i> = 2.5 GHz preliminary	-	-13	-	dB
	<i>f</i> = 4.0 GHz preliminary	-	-13	-	dB
	<i>f</i> = 5.0 GHz preliminary	-	-12	-	dB
SS data differential OFF isolation	<i>f</i> = 0.3 MHz preliminary	-	-100	-	dB
	<i>f</i> = 2.5 GHz preliminary	-	-80	-	dB
	<i>f</i> = 4.0 GHz preliminary	-	-80	-	dB
	<i>f</i> = 5.0 GHz preliminary	-	-80	-	dB
SS data differential crosstalk	<i>f</i> = 0.3 MHz preliminary	-	-90	-	dB
	<i>f</i> = 2.5 GHz preliminary	-	-35	-	dB
	<i>f</i> = 4.0 GHz preliminary	-	-32	-	dB
	<i>f</i> = 5.0 GHz preliminary	-	-32	-	dB
SS data propagation delay	preliminary	-	3.0	-	ns
SS data intra-pair skew	preliminary	-	10	-	ps
SS data inter-pair skew	preliminary	-	30	-	ps

HS data ON resistance	preliminary	-	9.0	-	$\Omega$
HS data ON resistance imbalance	preliminary	-	0.5	-	$\Omega$
HS data ON resistance flatness	V=0.0-1.0, VI=30mA	-	1.5	-	$\Omega$
HS data propagation delay	preliminary	-	0.6	-	Ns
HS data OFF isolation	preliminary	-	-48	-	dB
HS data crosstalk	preliminary	-	-30	-	dB
HS data 3dB bandwidth	preliminary	-	1200	-	MHz
SuperSpeed+ data rate	Depends on host and devices and cable loss link budget	-	-	10	Gbps
HiSpeed data rate	Depends on host and devices; SS data disabled	-	-	480	Mbps

Table 4: Typical Performance Characteristics



Figure 6: Typical SS data differential insertion loss

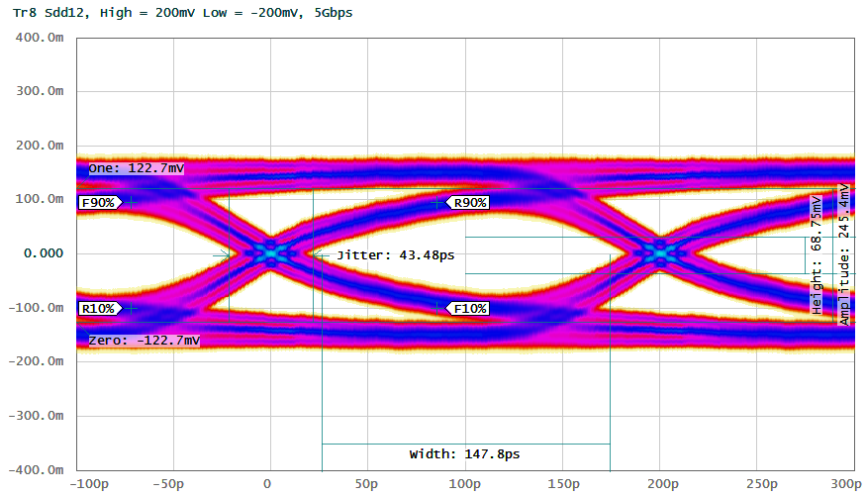


Figure 7: Simulated SS data 5Gbps eye diagram

## Overview

The USB-C-Switch is designed as a platform to simplify switching of multiple USB type-C ports. The switch is a bi-directional four-to-one or one-to-four multiplexer (mux) for USB-C connections. It is compatible with USB Type-C applications including SuperSpeed+ (10Gbps) and alternate modes (alt-mode). Supported alt-modes include HDMI, DisplayPort and digital audio.

At its core the switch is a passive 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 functional blocks for use in testing and debugging of USB Type-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 conduct a cable flip which reverses the apparent cable orientation to connected devices – automating the actions of unplugging, changing orientation and re-inserting a USB-C cable.

There are two sides to the USB-C-Switch: the Common Port (USB side) and the Mux Channel side. Using the BrainStem software API, the user can select which Mux Channel is routed to the Common Port. Only one Mux Channel can be active at a time. All signals are bi-directional, so either side can be connected to a host or device (see application examples above).

## Cable Flip

A key feature of the USB-C connector is its orientation independence. This makes the USB-C connector user-friendly, but complicates the development of devices using the USB Type-C standard. The USB Type-C specification and architecture makes determining connector orientation a responsibility of the active devices in the system. The orientation is defined by the cable or downstream device in the system; more specifically, by components inside of the USB-C male or plug side of a connection.

Figure 8 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.

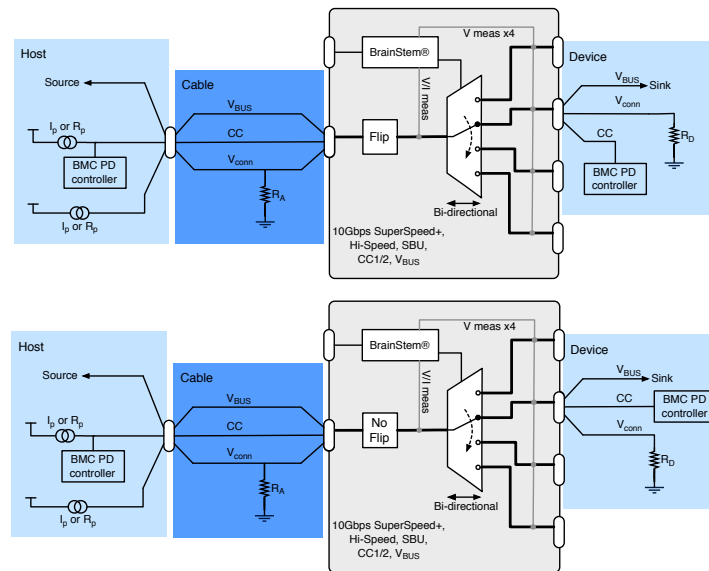


Figure 8: Flip and no-flip setting for full-featured cable and device

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 to connected devices that the orientation of their connection has reversed. It is incredibly import when testing any system with a USB type-C female connector, to ensure that any internal muxes are functioning and connected to the female connector as these are vital to the end-user “orientation-independent” experience of USB-C. 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 using the UOC cable, as a general rule, ensure that there is only one standard-compliant cable in the connection path between the USB host and USB device.

### 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 draws from the Control Port 5V supply and distributes power to the 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 features are enabled.

The KAC circuit does not provide any USB power-delivery (USB-PD), USB battery charge specification (BC1.2) or QuickCharge® to these inactive ports. KAC uses 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.

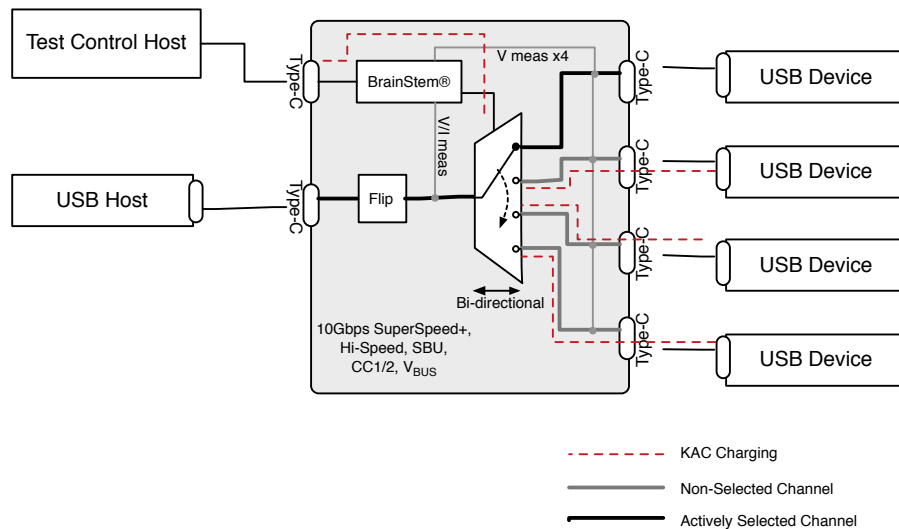


Figure 9: Typical example of KAC charging

## 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 the selected Mux Channel. In some cases, it may be advantageous for engineers 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 as a data connection, but to concurrently provide high rate charging through a separate charger. The USB-C-Switch has the unique and powerful feature of mux Split Mode to enable this level of flexibility and functionality.

Mux Default mode forces all USB-C signals to the selected Mux Channel.

Mux Split mode gives control over individual signal groups, allowing each group an independent Mux Channel assignment.

Signal groups under Split control assignment are:

- VBUS
- SSA (TX1+/-, RX1+/-)
- SSB (TX2+/-, RX2+/-)
- HSA (D+/-, Side A)
- HSB (D+/-, Side B)
- CC1
- CC2
- SBU1 and SBU2

In Split mode, VBUS 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 VBUS be assigned to only one Mux Channel. Caution should be used with multi-point VBUS assignments as it is possible to apply a VBUS voltage to a device that has not negotiated for high VBUS voltage and this could damage connected equipment.

When Split mode is enabled, USB-C-Switch will automatically disable the KAC circuit.

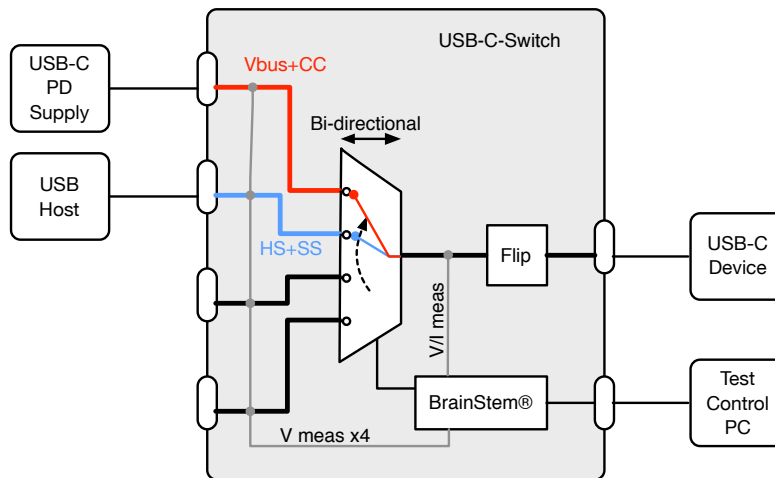


Figure 10: Adding USB-C PD charging capability to a legacy USB host output

**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.

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 system which provides simple high level APIs, a real-time embedded runtime engine and modular expandability. Functionality details unique to the USB-C-Switch are described in the following sections; refer to Table 8: Supported USB-C-Switch BrainStem Entity API Methods for a complete list of all available API functionality. All shortened code snippets are loosely based on the C++ method calls and meant to be psuedocode – Python and Reflex are virtually the same. Please consult the BrainStem Reference for implementation details.<sup>2</sup>

### 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.

```
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()
```

Saved Configurations	
USB Port Mode	Mux Configuration
Mux Split Mode	

### USB Entity

The `usb` entity provides a mechanism to control all USB functionality.

#### USB Channels

USB channels can be manipulated through the `usb` entity command to enable and disable USB data and  $V_{BUS}$  lines, measure current, measure  $V_{BUS}$  voltage, boost data line signals, and measure temperature.

The USBSwitch has a single USB channel at channel 0. It uses the mux entity to switch between one of 4 active mux channels.

Manipulating Hi-Speed data, SuperSpeed data, and  $V_{BUS}$  lines simultaneously for the USB channel can be done by calling the following methods with channel 0 as the index:

```
stem.usb.setPortEnable(0)
stem.usb.setPortDisable(0)
```

Manipulating Hi-Speed data and SuperSpeed data lines while not affecting the  $V_{BUS}$  lines simultaneously for a single port can be done by calling the following method with channel 0:

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

Manipulating just the USB 2.0 Hi-Speed data lines for a single port can be done by calling the following method with channel 0:

<sup>2</sup> See BrainStem software API reference at <https://acroname.com/reference/> for further details about all BrainStem API methods and information.

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

Manipulating just the USB 3.1 SuperSpeed data lines for a single port can be done by calling the following method with channel 0:

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

Manipulating just the USB  $V_{BUS}$  line for a single port can be done by calling the following method with channel 0:

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

To enable or disable or get the current status of the automatic orientation detection and connection functionality, the following methods can be called on the USB channel.

```
stem.usb.setConnectMode(0, 0|1)
stem.usb.getConnectMode(0, mode)
```

Enabling or disabling or getting the current status of just the Type-C CC lines for the USB channel can be done by calling the following methods.

```
stem.usb.setCC1Enable(0, 0|1)
stem.usb.setCC2Enable(0, 0|1)
stem.usb.getCC1Enable(0, value)
stem.usb.getCC2Enable(0, value)
```

Enabling or Disabling or getting the current status of the Type-C SBU lines for the USB channel can be done by calling the following methods.

```
stem.usb.setSBUEnable(0, 0|1)
stem.usb.getSBUEnable(0, value)
```

The USB  $V_{BUS}$  voltage, as well as the current consumed on  $V_{BUS}$ , can be read for the USB channel by calling the following methods with channel 0, where the second variable passed into the method is the location for the measurement result:

```
stem.usb.getPortVoltage(0,  $\mu$ V)
stem.usb.getPortCurrent(0,  $\mu$ A)
```

The Type-C CC line current and voltage can be read for the USB channel by calling the following methods with channel 0, where the second variable passed into the method is the location for the measurement result.

```
stem.usb.getCC1Voltage(0,  $\mu$ V)
stem.usb.getCC2Voltage(0,  $\mu$ V)
stem.usb.getCC1Current(0,  $\mu$ A)
stem.usb.getCC2Current(0,  $\mu$ A)
```

### USB Type-C Cable Flip

The USB-C-Switch can simulate a Type-C cable/connector flip by electrically switching the Side A and Side B  $CC/V_{CONN}$  and SBU lines, and swapping the USB data lines accordingly. This flip can be done, and checked, by calling the following methods:

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

The *setting* parameter is an integer that correlates to the following:

- 0 – normal
- 1 – flipped

### BrainStem Control Port

The USB-C-Switch has a dedicated control channel. This is a full-speed USB 2.0 connection for BrainStem interface only. No USB switch traffic can flow on this connection.

### USB Port Operational Mode

In addition to the individual port calls affecting specific functionality. The API includes a bitmapped method for setting specific port modes which allows for more granular control of the individual connections. The method takes a channel argument of 0 and a mode.

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

The value *mode* is 32-bit word, defined as the following:

Bit	Port Operational Mode Result Bitwise Description (1=Enabled, 0=Disabled)
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 <sub>BUS</sub> 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 5: Port Operational Mode Result Bitwise Description*

### **USB Port Operational State**

Setting port modes and affecting the port mod via individual API calls can affect the USB channel state. The API

includes a bitmapped method for getting the current port state. The method takes a channel argument of 0 and a mode.

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

The value *mode* is 32-bit word, defined as the following:

Bit	Port Operational State Result Bitwise Description (1=Enabled, 0=Disabled)
0	V <sub>BUS</sub> 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
10:11	Mux Channel orientation status
12:13	Reserved
14	CC Flip enable
15	Super-Speed Flip enable
16	SBU Flip enable
17:18	Daughter-Card status
20	Error Flag
21	Reserved
22	Reserved
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 6: Port Operational State Result Bitwise Description*

## Mux Entity

The USB-C-Switch provides the ability to switch the connection of the common type-C connector to one of four type-C switched channels. This is done with the `mux` entity with the desired channel as the parameter:

```
stem.mux.setChannel(channel)
```

where `channel` is an index 0-3.

### Mux Configuration

Default configuration of the mux is to switch all USB-C connections to a single channel. Changing the mux to split signals to different channel assignments, enabling Split mode. Default or Split can be enabled through the `mux` entity using the command:

```
stem.mux.getConfiguratiOn(config)
stem.mux.setConfiguratiOn(config)
```

where `config` value of

0 = Default mode; 1 = Split mode

### Mux Split Mode

USB-C connections can be individually assigned to separate mux channels through the `mux` entity using the command:

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

The value `splitMode` is 32-bit word, defined as the following:

Bit	Mux Split Mode State Result Bitwise Description (1=Enabled, 0=Disabled)
0:1	SBU1 11 – CH3 10 – CH2 01 – CH1 00 – CH0

2:3	SBU2 11 – CH3 10 – CH2 01 – CH1 00 – CH0
4:5	CC1 11 – CH3 10 – CH2 01 – CH1 00 – CH0
6:7	Reserved
8:9	CC2 11 – CH3 10 – CH2 01 – CH1 00 – CH0
10:11	Reserved
12:13	HS Side A Data 11 – CH3 10 – CH2 01 – CH1 00 – CH0
14:15	HS Side B Data 11 – CH3 10 – CH2 01 – CH1 00 – CH0
16:17	SS Lane 1 Data 11 – CH3 10 – CH2 01 – CH1 00 – CH0
18:19	SS Lane 2 Data 11 – CH3 10 – CH2 01 – CH1 00 – CH0
20	V <sub>BUS</sub> enable CH0
21	V <sub>BUS</sub> enable CH1
22	V <sub>BUS</sub> enable CH2
23	V <sub>BUS</sub> enable CH3
24:31	Reserved

Table 7: Mux Split Mode Result Bitwise Description

## 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	
system[0]	slotSize	
	save	
	reset	
	setLED	
	getLED	
	getInputVoltage	
	getVersion	
	getModuleBaseAddress	
	setHBInterval	
	getHBInterval	
	getModule	
	getSerialNumber	
timer[0-8]	getRouter	
	getModel	
	getExpiration	
	setExpiration	
usb[0]	getMode	
	setMode	
	setPortEnable	
	setPortDisable	

	setDataEnable	
	setDataDisable	
	setHiSpeedDataEnable	
	setHiSpeedDataDisable	
	setSuperSpeedDataEnable	
	setSuperSpeedDataDisable	
	setPowerEnable	
	setPowerDisable	
	getPortVoltage	
	getPortCurrent	
	getPortMode	
	setPortMode	
	getPortState	
	getDownstreamBoostMode	
	setDownstreamBoostMode	
	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
--	------------	--------------

Table 8: Supported USB-C-Switch BrainStem Entity API Methods<sup>3</sup>

---

<sup>3</sup> See BrainStem software API reference at <https://acroname.com/reference/> for further details about all BrainStem API methods and information.

### Pinouts

The USB-C-Switch uses standard USB pin outs for the type-C female receptacles shown in Figure 11. The side-A and side-B USB HS D+ and D- are separately passed through the USB-C-Switch.


A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	 Receptacle (Front View)
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	

Figure 11: USB type-C receptacle pin out

### LED Indicators

The control side of the USB-C-Switch has 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/Yellow	Indicates system is powered, and indicates heartbeat status. Pulses at a rate determined by the system heartbeat rate to indicate an active BrainStem link.
Side A USB Status	Green	Indicates Mux Channel selection. Disabled when Split mode is enabled.
Side B USB Status	Green	
Channel 0 Status	Blue	
Channel 1 Status	Blue	
Channel 2 Status	Blue	
Channel 3 Status	Blue	

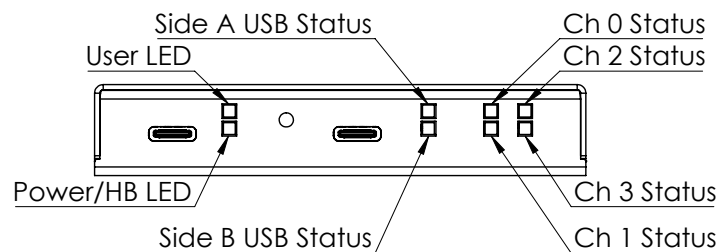
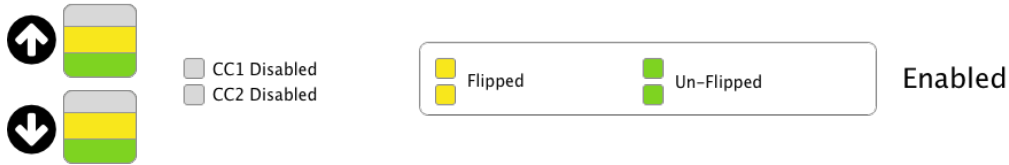


Figure 12: LED positions

## Connection Mode Manual



## Connection Mode Auto

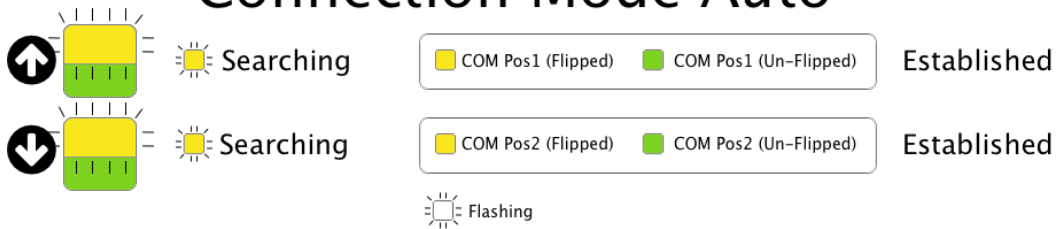


Figure 13: LED status

## 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.

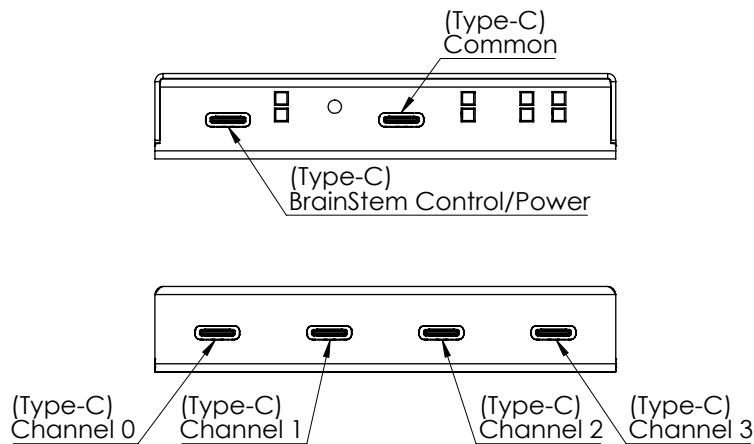


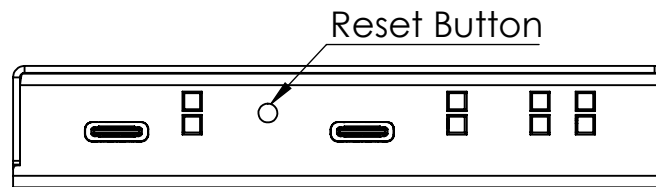
Figure 14: USB-C connector names

## Power Input

Power for the USB-C-Switch is provided by the  $V_{BUS}$  line on the BrainStem 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

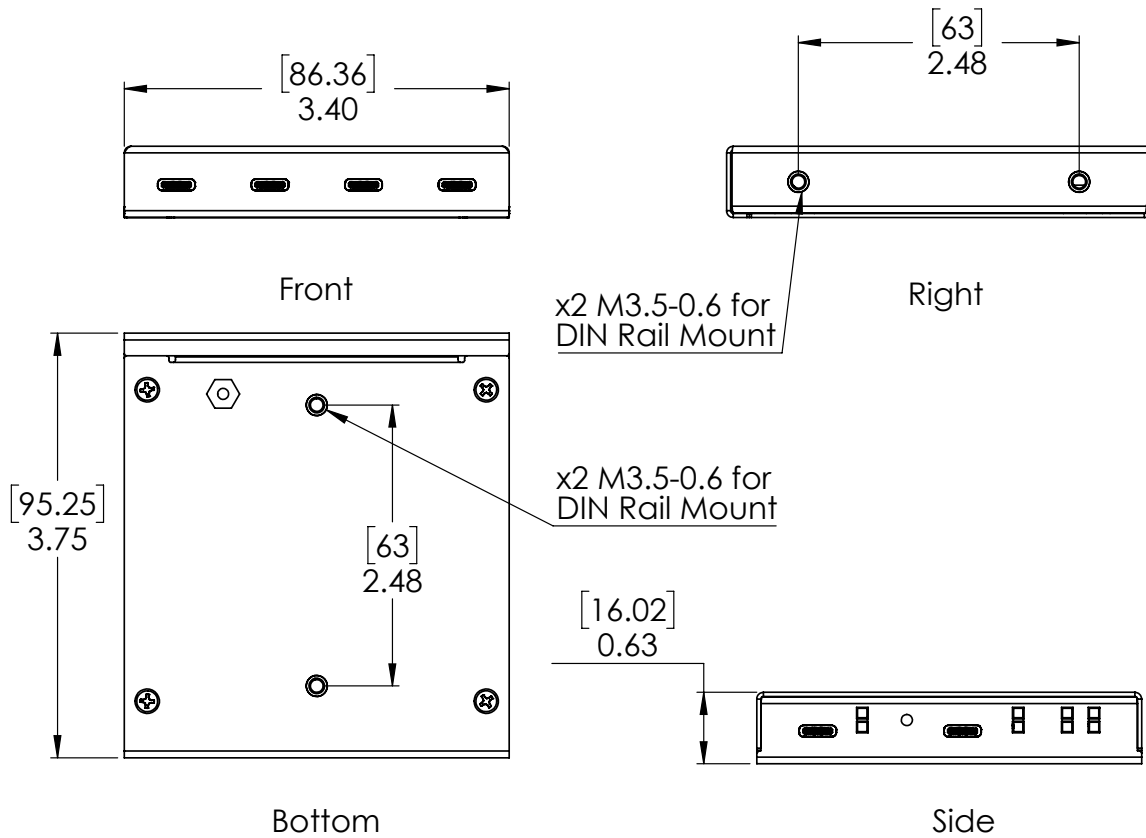
The USB-C-Switch can be reset to factory default settings using the reset button on the control side. Pressing the reset button once will restart the USB-C-Switch as if it had been power cycled. To restore factory default settings, press the Reset Button two times within 5 seconds.



*Figure 15: Reset button location*

**Mechanical**

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



DIMENSIONS: IN[MM]  
SCALE: 2:3

Figure 16: USB-C-Switch Mechanical Dimensions

## DIN Rail Mounting

DIN rail mounts have been designed into the USB-C-Switch case with an appropriate clip as often used for industrial control equipment. Mounting clip hardware is not included with the USB-C-Switch. The mounting holes are compatible with many widely available “small” DIN rail mounting clips, and Acroname part number C31-DINM-1. The USB-C-Switch can be mounted in two positions as shown in figure Figure 17.

*Warning: Care should be taken to only use M3.5 8mm long screws. Longer screws will cause irreparable damage to the USB-C-Switch.*

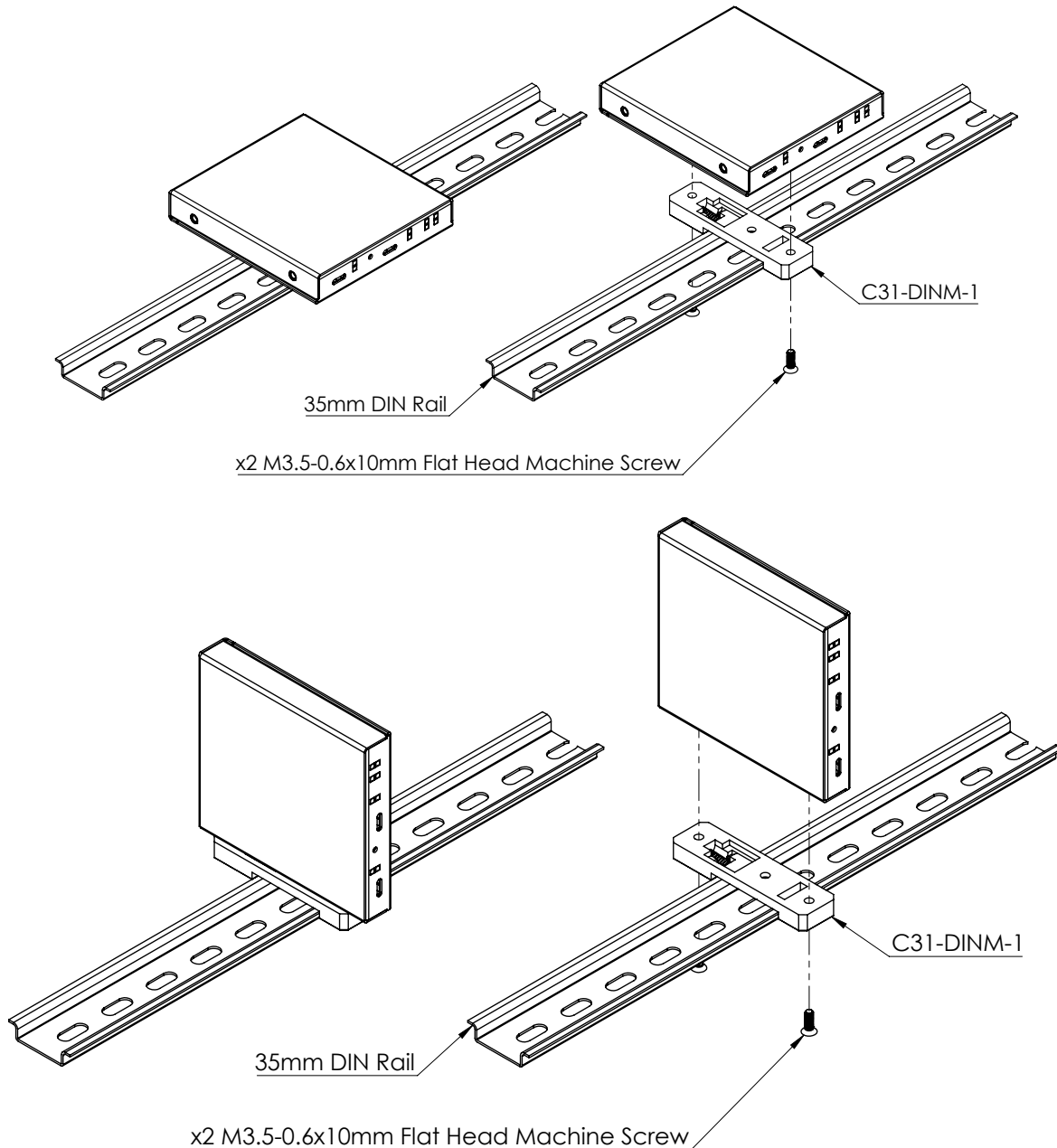


Figure 17: USB-C-Switch DIN Rail mounting

## 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.

## 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