Overview

The USBHub2x4 is a 4-port software-programmable USB 2.0 (480Mbps) hub that is designed for demanding industrial environments where advanced control and monitoring of USB ports is required. This is very useful in testing or development environments where standard "always-on" behavior of a consumer-grade USB hub is not desirable.

Software control of the USBHub2x4 is established and maintained over the selected one of the two available host-facing ports.

The USBHub2x4 can be used to enable/disable individual USB ports, measure current or voltage on downstream USB ports, set programmable current limits, set USB charging protocol behavior and otherwise automate USB port behaviors in development and testing.

Typical applications include:

- USB device manufacturing
- USB device validation and development
- Functional testing
- Camera control
- Battery charging
- USB device resets
- USB monitoring
- Sequential firmware load/updates

Features

- Individually enable/disable any of 4 downstream ports
- Data and power lines can be separately enabled for each downstream port
- Measure voltage and current on each downstream port
- Set programmable current limits for each downstream port (500mA to 2.5A)
- Automatic or programmed selection for either of 2 host port connections
- All ports support USB link speeds up to 480Mb/s
- Selectively enable USB charging mode behaviors: SDP (Standard Downstream Port) or CDP (Charging Downstream Port) modes[1]
- Deliver up to 2.5A per port (in CDP mode)
- Set enumeration delay for discovery of attached downstream devices
- Boost USB upstream and downstream data signal levels
- DIN-rail mountable
- Certified to withstand +/- 30kV ESD strikes (IEC6100-4-2 level 4)

Description

The USBHub2x4 gives engineers advanced flexibility and configurability over USB ports in testing and development applications.

Each downstream USB channel implements separately and independently switched data lines and 2500mA current-limited power lines. USB power and data can be independently disconnected for advanced USB testing applications. Pin interfaces are protected against reverse polarity and over-voltage, and connections are designed to operate from 0°C to 50°C ambient with no external cooling or fans.

Each USBHub2x4 is uniquely addressable and controllable from a host PC via the selected USB host input. Acroname's BrainStem™ link is then established over the USB input and allows a connection to the on-board controller in the USBHub2x4. USBHub2x4 can be controlled via a host running BrainStem APIs or alternately, it can operate independently by running locally embedded, user-defined programs based on Acroname's BrainStem Reflex language.

[1] See http://www.usb.org/developers/docs/devclass_docs/ under the category Battery Charging for full details.

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, V_{supply}	6.0	26.0	V
V_{supply} current	0.0	14.0	A
Voltage on V_{bus} inputs	0.0	24.0	V
Voltage on V_{bus} outputs	0.0	6.0	V
Voltage on any USB D+/D- inputs and outputs	-0.3	5.3	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
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 and contact discharge	0.0	-	±30	kV

Table 2: Handling Ratings

Recommended Operating Ratings

Specifications are valid at 25°C unless otherwise noted. Intended for indoor use only.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Unit
Input Voltage, V_{supply}		9.0	12	24.0	V
Input Current, I_{supply}		0.15	-	11.0	A
Voltage on V_{bus} inputs and outputs		4.5	5.0	5.5	V
Relative Humidity Range	Non-Condensing	5	-	95	%RH

Table 3: Recommended Operating Ratings

Block Diagram

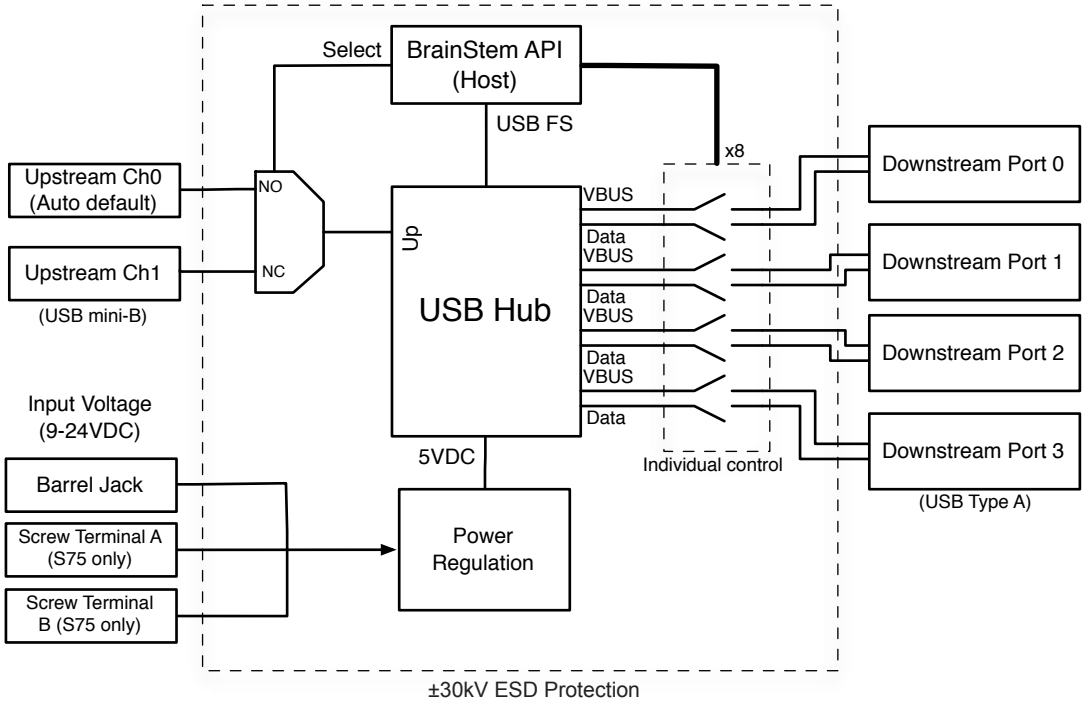


Figure 1: USBHub2x4 Block Diagram

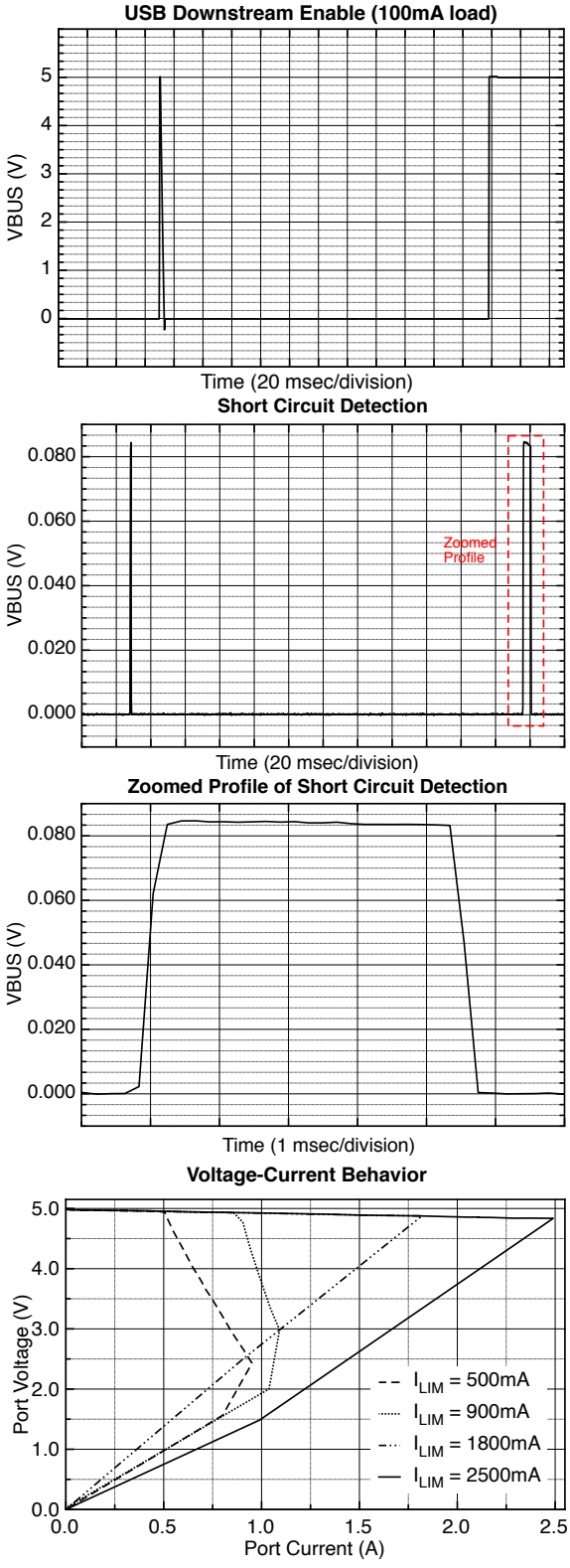
Typical Performance Characteristics

Specifications are valid at 25°C unless otherwise noted. Indoor application 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	Unit
USB Downstream (V_{bus})		4.5	5.0	5.5	V
USB Downstream Current	I_{LIM}=2.5A	0.0	-	2.5	A
System Efficiency	@12.0V input, nominal 6.5A load[2]	84	-	86	%
Current Measurement Range		6.4	-	2500	mA
Current Measurement Resolution		-	9.76	-	mA
Current Measurement Accuracy	I_{LIM} not exceeded		±2		%
V_{bus} Voltage Measurement		0.0	-	5.5	V
V_{bus} Voltage Resolution		-	1.2	-	mV
V_{bus} Output Rise Time	I_{LIM} = 1.0A	-	-	1.0	ms
V_{supply} Measurement Resolution		-	8	-	mV
Selectable Current Limits I_{LIM}	I_{LIM} = 500mA I_{LIM} = 900mA I_{LIM} = 1000mA I_{LIM} = 1200mA I_{LIM} = 1500mA I_{LIM} = 1800mA I_{LIM} = 2000mA I_{LIM} = 2500mA	- - - - - - - -	480 850 950 1130 1400 1720 1910 2370	500 900 1000 1200 1500 1800 2000 2500	mA
Short Circuit Response Time	Time from detection of short to current limit applied.	-	-	1.5	µs
Short Circuit Detection Time	Time from detection of short to port power switch disconnect.	-	-	6.0	ms
USB Downstream V_{bus} Current Supply (SDP mode)	USB 2.0 data lines disabled or no USB host present	-	-	100	mA
USB Downstream V_{bus} Current Supply (SDP mode)	USB 2.0 data lines enabled and USB host must be present	-	-	500	mA
USB Downstream V_{bus} Current Supply (CDP mode)	USB 2.0 data lines must be enabled	-	-	1500	mA

Table 4: Electrical Characteristics

[2] 6.5A selected as representative load based on 4 USB downstream devices running in CDP mode consuming approximately 1.5A each.



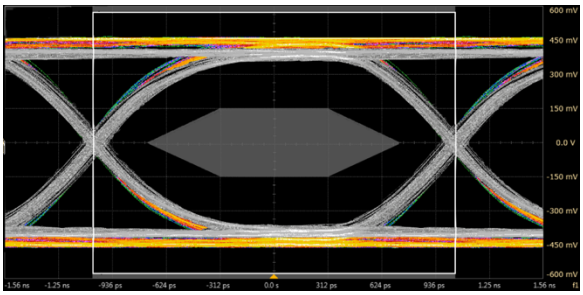


Figure 2: Upstream USB Eye diagram through USB Mini-B to host with 0.3m cable. Boost 0% in greyscale; Boost 12% in color.

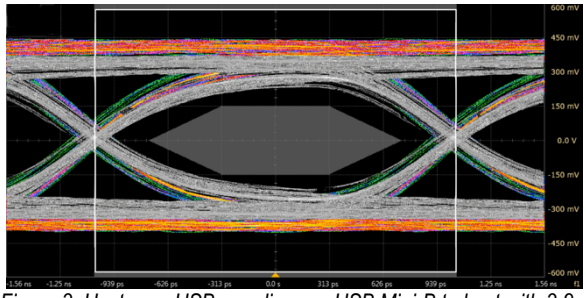


Figure 3: Upstream USB eye diagram USB Mini-B to host with 3.2m cable. Boost 0% in greyscale; Boost 12% in color.

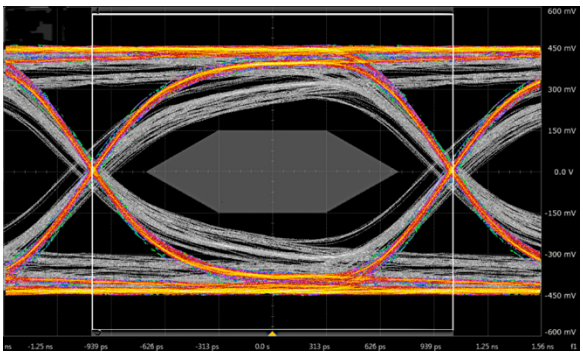


Figure 4: Downstream USB eye diagram USB A to device. 3.2m cable in greyscale; 0.3m cable in color.

Module Hardware and Software Default Values

The USBHub2x4 leverages a hardware-specific subset of BrainStem Entity implementations. The `aUSBHub2x4.h` C++ header file includes macro definitions for many parameters specific to the USBHub2x4. Table 5: USBHub2x4 Hardware and Software Default Values provides an overview of these values.

Parameter	Index	Macro Name or Implemented Options	Notes
Module Definitions:			
Module Base Address	6	`aUSBHUB2X4_MODULE_ADDRESS`	
Router Base Address	6		
Entity Class Definitions:			
`timer` Entity Quantity	8	`aUSBHUB2X4_NUM_TIMERS`	
`usb` Entity Quantity	1	`aUSBHUB2X4_NUM_USB`	
`store` Entity Quantity	2	`aUSBHUB2X4_NUM_STORES`	
`system` Entity Quantity	1		
`app` Entity Quantity	4	`aUSBHUB2X4_NUM_APPS`	
`pointer` Entity Quantity	4	`aUSBHUB2X4_NUM_POINTERS`	

Table 5: USBHub2x4 Hardware and Software Default Values[3]

[3] Refer to `aUSBHub2x4.h` within the BrainStem Development Kit download for actual file.

Device Drivers

USBHub2x4 leverages operating system user space interfaces that do not require custom drivers for operation on modern operating systems.

Some older operating systems may require the installation of a BrainStem USB driver to enable software control. Installation details on installing USB drivers can be found within the BrainStem Development Kit under the "drivers" folder. For example, Windows 7 the supplied INF to communicate with BrainStem USB devices.

Capabilities and Interfaces

The USBHub2x4 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 USBHub2x4 are described in the following sections. A complete list of available API functionality specific to USBHub2x4 is listed in Table 10. All shortened code snippets are loosely based on the C++ method calls and meant to be Psuedocode like – Python and Reflex are virtually the same. Please consult the online BrainStem Reference for implementation details[4].

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.

Serial Number

Every USBHub2x4 is assigned a unique serial number at the factory. This facilitates an arbitrary number of USBHub2x4 devices attached to a host computer. The following method call can retrieve the unique serial number for each device.

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

Module Default Base Address

BrainStems are designed to be able to form a reactive, extensible network. All BrainStem modules come with a default network base address for identification on the BrainStem network bus. The default module base address for USBHub2x4 is factory-set as 6, and can be accessed with:

```
stem.system.getModule(module)
```

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 USBHub2x4 away from the factory default settings. Saving system settings preserves the settings as the new default. Most changes to system settings require a save and reboot before taking effect. For example, upstream and downstream USB Boost settings will not take effect unless a system save operation is completed, followed by a reset or power cycle. Use the following command to save changes to system settings before reboot:

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

Pressing the reset button two times within 5 seconds will return all settings to factory defaults: all ports' data and power enabled, CDP mode, enumeration delay of 0, 2500mA current limit.

Saved Configurations	
Software Offset	I2C Rate
Router Address	Port Enumeration Delay
Boot Slot	Downstream Boost
Port Mode (SDP, CDP) – each port	Current Limit – per port
Upstream Boost	Port state (data and power)

Table 6: Saved Parameters

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

USB Entity

The `usb` entity provide a mechanism to control all functionality for the upstream and downstream USB ports.

USB Downstream Channel Control

Downstream 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.

Manipulating Hi-Speed data and V_{bus} lines simultaneously for a single port can be done by calling the following methods with channel in [0-3] being the port index:

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

Manipulating data lines while not affecting the V_{bus} lines simultaneously for a single port can be done by calling the following methods with channel [0-3]. The follow methods provide equivalent functionality; the two methods are offered for compatibility with other products.

```
stem.usb.setDataEnable(channel)
stem.usb.setDataDisable(channel)
stem.usb.setHiSpeedDataEnable(channel)
stem.usb.setHiSpeedDataDisable(channel)
```

Manipulating just the USB V_{bus} line for a single port can be done by calling the following method with channel [0-3]:

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

To affect multiple ports and lines simultaneously, see `usb.setHubMode()` later in this section.

Note that transitions between power and data enable states where power is enabled and only data is changing require the USBHub2x4 to toggle V_{bus} power. This appears as

USB Downstream Measurements

The USB V_{bus} voltage, as well as the current consumed on V_{bus}, can be read for each channel by calling the following methods with channel [0-3], where the second variable passed into the method is the location for the measurement result:

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

USB Downstream Current Limit

Current-limit settings can be accessed for each port by calling the following methods with channel [0-3]. The second variable passed into the method is either the set value or the write location of the result. The set value can be any value and will be rounded down by the USBHub2x4 to allowed values listed in Table 4. When a connected downstream device attempts to consume more current than the set current limit, the USBHub2x4 will enter a constant-current mode, reducing the V_{bus} voltage until the current limit set point is reached.

```
stem.usb.getPortCurrentLimit(channel,μA)
stem.usb.setPortCurrentLimit(channel,μA)
```

The current-limiting behavior follows the USB BC1.2 specification which allows for many different behaviors. The USBHub2x4 has two stages of current-limiting. When a downstream device consumes current higher than the programmed current limit, the hub will enter a “constant current” mode and is indicated in the `getPortState()` bitfield with the constant current bit. In the constant current mode, the V_{bus} voltage will be reduced to attempt maintain a constant current at the set current limit. The time and amount of voltage reduction and maximum allowed current draw depends on the current limit set point. Please see figures above for expected V-I characteristics.

As the V_{bus} voltage is reduced, if the device continues to increase its current draw (reduce it’s effective resistance), the USBHub2x4 will “trip off” by disabling the V_{bus} and high-speed data lines. This state is indicated with the error bit in the `getPortState()` bitfield. The Channel X Power error LED will also illuminate when this error occurs. See the LED Indicators section for details.

USB Downstream Enumeration

The enumeration state and speed of each downstream port can be read with

```
stem.usb.getDownstreamDataSpeed(ch,speed)
```

with ch in [0-3] and speed values retuned as:

Value	Hub Downstream Speed Descriptions
0	No device enumerated
1	Hi-Speed device enumerated

Table 7:Hub downstream speed value description

USB Downstream Operational Mode

The USB port operational mode controls the behavior of each downstream port’s charging behavior. Each port can be setup to support different modes in the USB Battery Charge Specification 1.2 (BC1.2). Standard Downstream Port (SDP) mode will cause BC1.2 compliant or older USB devices to consume 500mA or less. Configuring a port as a Charging Downstream Port (CDP) will cause the hub signal to downstream devices that devices may consume up to 5A, the maximum allowed by BC1.2. If there is no upstream USB host

connected to the hub, downstream ports set to CDP will behave as Dedicated Charging Ports (DCP).

The actual current consumed by the device is controlled by the downstream device and not the USBHub2x4. Devices which are not compliant with BC1.2 or the previous USB power specifications may draw more current than specified above.

The operational mode is set or read by calling the methods:

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

Available options for Downstream Operational Mode are:

Value	Hub Port Mode Descriptions
0	Standard downstream port (SDP)
1	Charging downstream port (CDP)

Table 8: Hub port mode value description

USB Downstream Enumeration Delay

Once a USB device is detected by the USBHub2x4 it is possible to delay its connection to an upstream host computer and subsequent enumeration on the USB bus. The enumeration delay can mitigate or eliminate host kernel instabilities by forcing devices to enumerate in slow succession, allowing a focus on validation of drivers and software. The enumeration delay is configured in milliseconds, representing the time delay between enabling each successive downstream port from 0 to 3. Enumeration delay is applied when the hub powers on or when a new upstream connection is made.

```
stem.usb.setEnumerationDelay(delay)
stem.usb.getEnumerationDelay(delay)
```

USB Boost Mode

Boost mode increases the drive strength of the USB 2.0 Hi-Speed data signals (power signals are not changed). Boosting the data signal drive strength may help to overcome connectivity issues when using long cables or connecting through relays, "pogo" pins or other adverse conditions. This setting is applied after a `system.save()` call and reset or power cycle of the hub. The system setting is persistent until changed or the hub is hard reset. After a hard reset, the default value of 0% boost is restored. A hard reset is done by pressing the “Reset” button on the back of the hub two times within 5 seconds while the hub is powered.

Boost mode can be applied to both the upstream and downstream USB ports with the follow methods:

```
stem.usb.getDownstreamBoostMode(setting)
stem.usb.setDownstreamBoostMode(setting)
stem.usb.getUpstreamBoostMode(setting)
stem.usb.setUpstreamBoostMode(setting)
```

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

Value	Hub Boost Mode Descriptions
0	Normal drive strength
1	4% increase in drive strength
2	8% increase in drive strength
3	12% increase in drive strength

Table 9: Hub boost mode value descriptions

USB Hub Upstream Channels

The USBHub2x4 is perfect for environments where multiple devices need to be shared or switched between two host computers using two host (upstream) connections via USB mini-B connectors. The upstream connection can be automatically detected or specifically selected using the following methods:

```
stem.usb.getUpstreamMode(mode)
stem.usb.setUpstreamMode(mode)
```

The *mode* parameter can be defined as the following:

Value	Hub Upstream Mode Descriptions
0	Force upstream port 0 to be selected
1	Force upstream port 0 to be selected
2	Automatically detect upstream port

Table 10: Hub upstream mode value descriptions

Predefined C++ macros for these can be found in aProtocoldef.h, and Python’s built-in help interface.

The default operational mode is to auto detect which upstream USB port is selected. Automatic detection uses the presence of V_{bus} on the USB type-B upstream connector to determine presence of a host. i.e. if a host is connected to the Up0 port, then channel 0 will be selected as the upstream USB connection (see Figure 3 for more detail).

Note that in the USBHub2x4, the BrainStem API and control communication path is only available through the actively selected upstream port.

USB Hub Upstream State

The USBHub2x4 can provide status information on which upstream port is actively selected as data path to the downstream ports:

```
stem.usb.getUpstreamState()
```

This command returns a 32-bit value which indicates:

Value	Hub Upstream State Descriptions
0	Upstream port 0 is actively selected
1	Upstream port 1 is actively selected
2	No upstream port is selected

Table 11: Hub upstream state value descriptions

USB Hub Operational Mode

In addition to targeting individual downstream USB ports, a bit-mapped hub state interface is also available. This interface allows the reading or setting of all USB downstream ports in one functional call.

```
stem.usb.getHubMode(mode)
```

```
stem.usb.setHubMode(mode)
```

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

Bit	Hub Operational Mode Bitwise Description
0	USB Channel 0 USB Hi-Speed Data Enabled
1	USB Channel 0 USB V_{bus} Enabled
2	USB Channel 1 USB Hi-Speed Data Enabled
3	USB Channel 1 USB V_{bus} Enabled
4	USB Channel 2 USB Hi-Speed Data Enabled
5	USB Channel 2 USB V_{bus} Enabled
6	USB Channel 3 USB Hi-Speed Data Enabled
7	USB Channel 3 USB V_{bus} Enabled
8:31	Reserved

Table 12: Hub Operational Mode Result Bitwise Description

USB Port State

Each downstream port reports information regarding its operating state represented in bit-packed results from:

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

where *channel* can be [0-3], and the value *status* is 32-bit word, defined as the following:

Bit	Port State: Result Bitwise Description
0	USB V_{bus} Enabled
1	USB2 Data Enabled
2:18	Reserved
19	USB Error Flag
20	USB2 Boost Enabled
21:22	Reserved
23	Device Attached
24	Constant Current Mode
25:31	Reserved

Table 13: Port State: Result Bitwise Description

USB Port Error Status Mapping

Error states for all downstream ports are bit-packed in 32-bit words available from:

```
stem.usb.getPortError(channel)
```

where *channel* is [0-3].

Errors can be cleared on each individual channel by calling the following method:

```
stem.usb.clearPortErrorStatus(channel)
```

Calling this command clears the port-related error bit flags (see Table 9) in the port error state. Global bits for hub errors cannot be cleared by this command.

Details about the port error status 32-bit word are as follows:

Bit	Port Error Status Bitwise Description
0	USB port current limit exceeded
1	USB port back-drive condition detected
2	Reserved
3	Hub overtemperature condition
4:31	Reserved

Table 14: Port Error Status Result Bitwise Description

USB System Temperature

The temperature of the USB subsystem in the USBHub2x4 can be measured with:

```
stem.temperature(0).getTemperature(μC)
```

where temperature is in micro-degrees Celsius.

USBHub2x4 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 USBHub2x4 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	
	setSleep	
	setBootSlot	
	getBootSlot	
	getInputVoltage	
	getInputCurrent	
	getVersion	
	setHBInterval	
	getHBInterval	
	getModule	
	getSerialNumber	
	getModel	
temperature[0]	getTemperature	
timer[0-8]	getExpiration	
	setExpiration	
	getMode	
	setMode	
usb[0]	setPortEnable	Channels 0-3
	setPortDisable	Channels 0-3
	setDataEnable	Channels 0-3
	setDataDisable	Channels 0-3
	setHiSpeedDataEnable	Channels 0-3
	setHiSpeedDataDisable	Channels 0-3
	setPowerEnable	Channels 0-3
	setPowerDisable	Channels 0-3
	getPortVoltage	Channels 0-3
	getPortCurrent	Channels 0-3
	getPortCurrentLimit	Channels 0-3
	setPortCurrentLimit	Channels 0-3
	setPortMode	Channels 0-3
	getPortMode	Channels 0-3
	getDownstreamDataSpeed	Channels 0-3
	getHubMode	

Entity Class	Entity Option	Variable(s) Notes
	setHubMode	
	getPortState	Channels 0-3
	getPortError	
	getEnumerationDelay	
	setEnumerationDelay	
	clearPortErrorStatus	
	getUpstreamMode	
	setUpstreamMode	
	getUpstreamState	
	getUpstreamBoostMode	
	setUpstreamBoostMode	
	getDownstreamBoostMode	
	setDownstreamBoostMode	
Pointer[0-3]	getOffset	
	setOffset	
	getMode	
	setMode	
	getTransferStore	
	setTransferStore	
	initiateTransferToStore	
	initiateTransferFromStore	
	getChar	
	setChar	
	getShort	
	setShort	
	getInt	
	setInt	
App[0-3]	execute	

Table 15: Supported USBHub2x4 BrainStem Entity API Methods[5]

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

LED Indicators

Built into the board are a number of LED indicators to assist in system troubleshooting using the USBHub2x4.

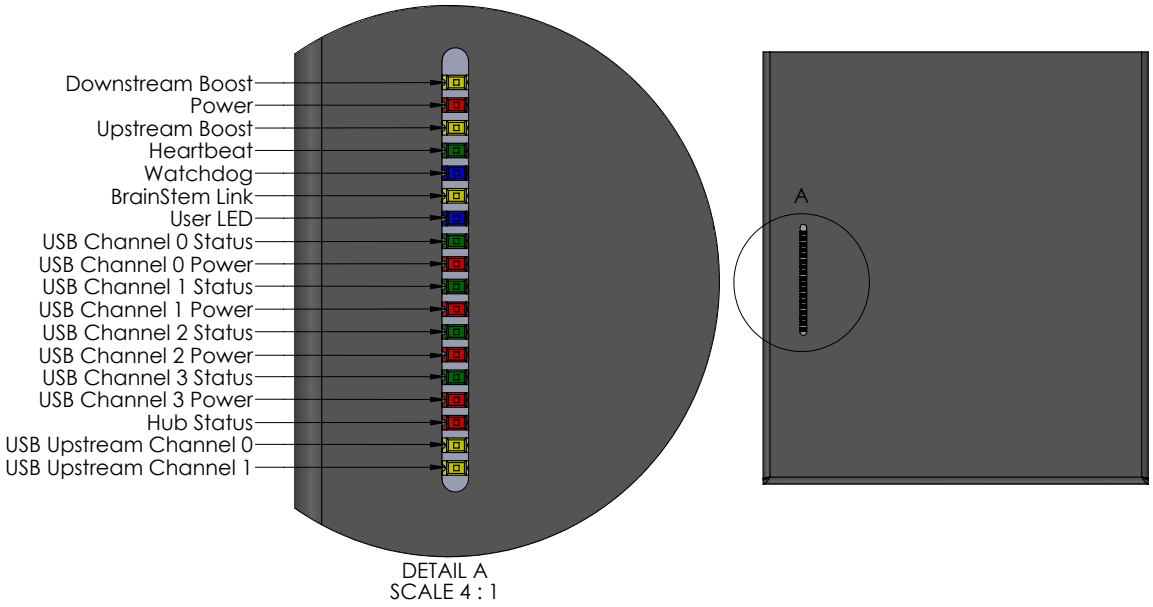


Figure 5: USBHub2x4 LED Indicators

The BrainStem Link LED will illuminate when the BrainStem USB interface is created on a host computer.

The Heartbeat indicator informs the user when communication is occurring with the BrainStem module, including a periodic heartbeat signal and response. Additional details on Heartbeats can be found in the BrainStem Terminology section of the Reference Manual.

The Logic Power indicator shows that a 3.3V voltage regulation system is up and running properly.

The User LED is a software controllable indicator accessed via the System BrainStem Entity. Detailed information can be found in the System Entity section of the reference manual.

Each downstream USB connection has LED indicators for status and power. The red LED labeled USB Channel X Power indicates an error on USB power (V_{bus}) such as overcurrent. This LED will automatically turn-off approximately 5 seconds after the error condition occurs. Additionally, a green LED, labeled USB Channel X Status, indicates whether the downstream device has enumerated on the host computer.

The Hub Status indicator illuminates when the USB hub communicates with a host computer.

Using Multiple Hosts with USBHub2x4

The two upstream-facing host ports can be connected to two different host computers. Due to limitations of USB specification and architecture, only one host computer can access downstream USB ports at any time. Through the BrainStem APIs, the upstream port used can be specifically selected, or the system can automatically select the upstream port.

The BrainStem controller connection is available only through the actively selected host port. Because of this, it is important to keep in mind that when a host upstream connection is changed, the software connection to the BrainStem module should follow the upstream connection appropriately.

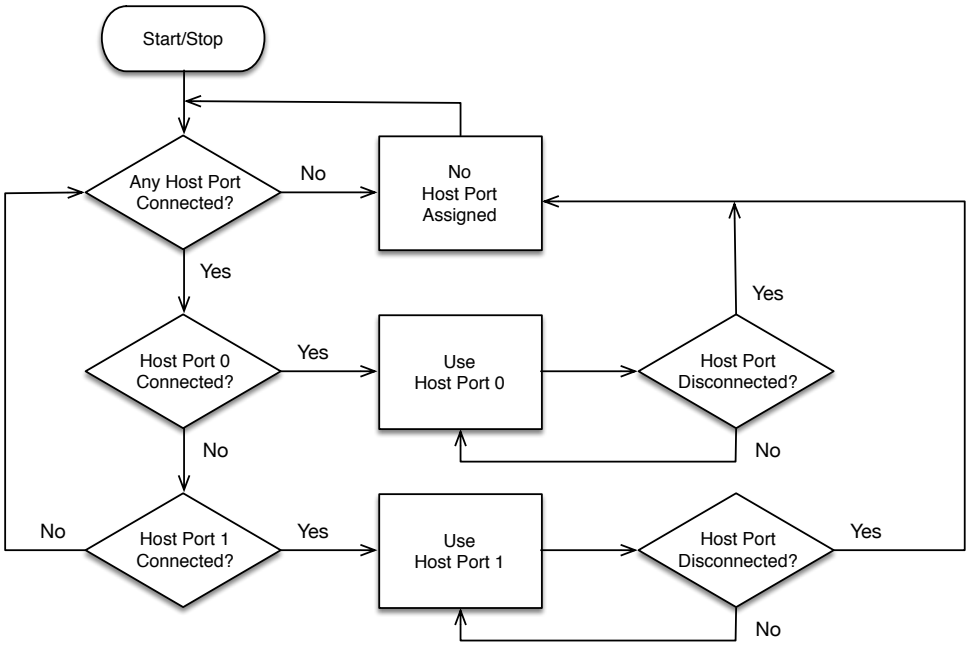


Figure 6: Determining connection used for BrainStem communications

Mechanical

Dimensions are shown in inches [mm]. 3D CAD models are available through the USBHub2x4 product page's Downloads section

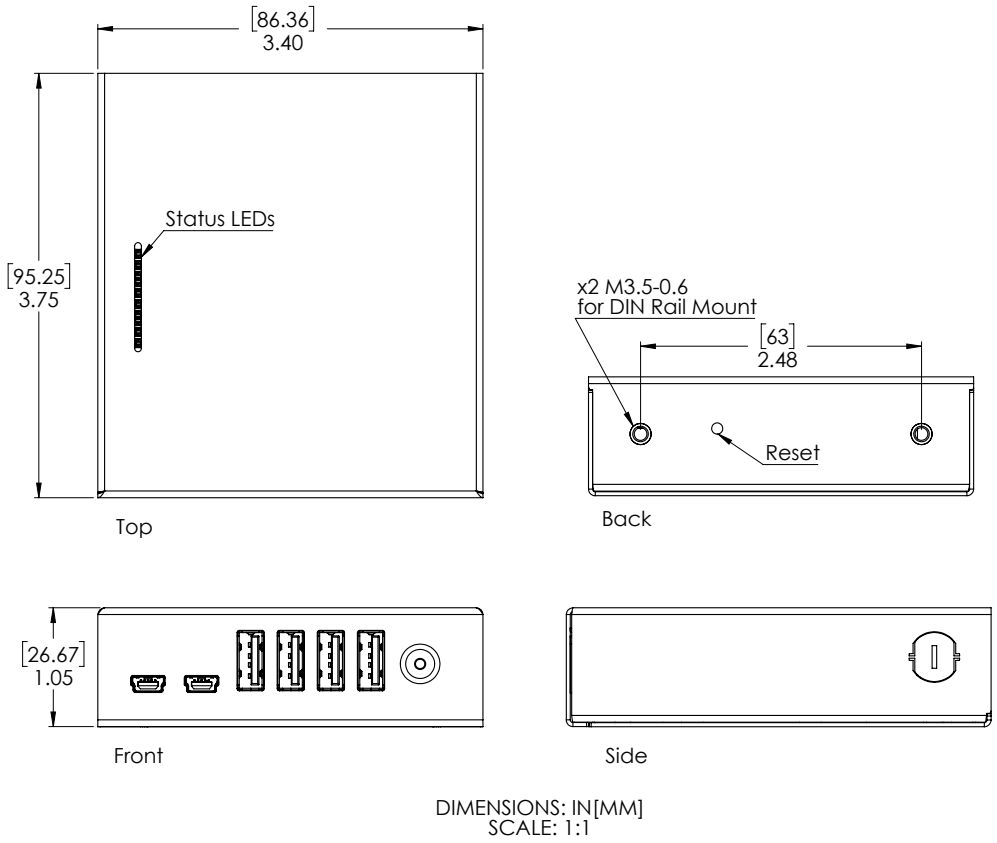


Figure 7: USBHub2x4 Mechanical

Input Power Connections

The USBHub2x4 can be powered through the DC barrel-jack on the front of the unit.

The DC barrel-jack is a standard 5.5mm outside diameter, 2.5mm inside diameter, 9.5mm mating length connector. Many manufacturers make compatible mating plug connectors; one example is the DC barrel plug from CUI: part number PP3-002B (https://www.cui.com/product/resource/pp3-002b.pdf).

DIN Rail Mounting

DIN rail mounting provisions have been designed into the USBHub2x4 case. Holes for a DIN rail clip/adapter are provided to allow mounting of the USBHub2x4 to standard DIN rails. Mounting clip hardware is available separately in a kit from Acroname: part number C31-DINM-1. The diagrams below illustrate USBHub2x4 mounted in two orientations:

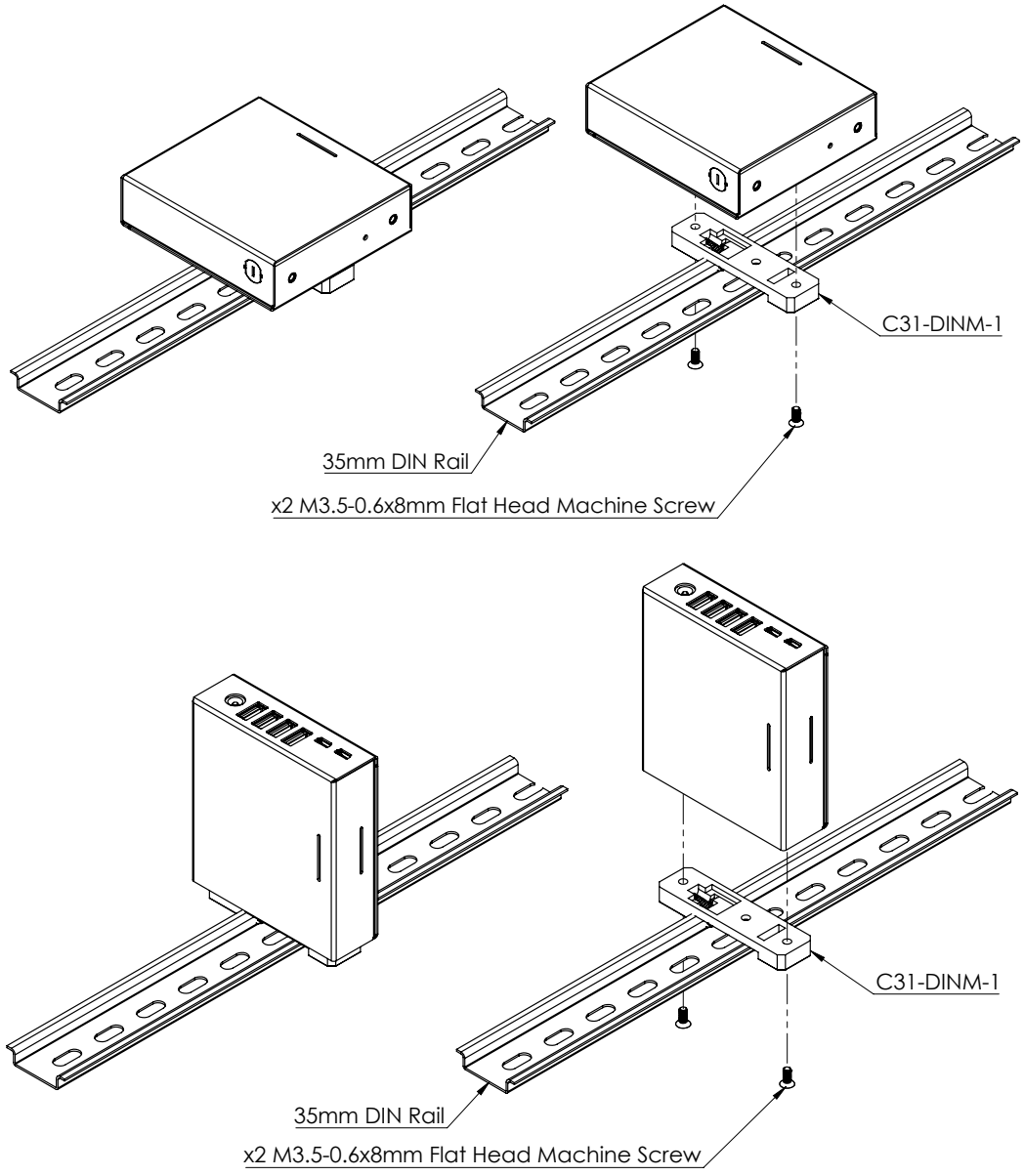


Figure 8: USBHub2x4 DIN Rail Mount

FCC Compliance Statement

Note: 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
1.0	April 2015	MJK	Initial Revision
1.1	September 2015	JTD	Reformatted. Added Entity Section Specifics, DIN rail mounting
1.2	November 2015	JTD	Updated DIN rail mounting
1.3	December 2015	JTD	Updated ESD rating
1.4	January 2016	JLG	Typographical and formatting fixes
1.6	February 2016	JLG	Update part number for DIN rail mount; FCC Compliance; add block diagram
1.7	February 2016	JLG	Update Electrical Characteristics table
1.8	March 2016	JTD	Updated CAD to v2
1.9	September 2016	JTD	Updated formatting
1.10	October 2016	LCD	Updated Overview, Features, Description sections
1.11	March 2017	JTD	Updated DIN mount screw spec
1.12	April 2018	RMN	Swapped hubState for portState
1.13	November 2018	LCD	Updated for BrainStem API changes
1.14	July 2020	ACRO	Formatting update
1.15	Jan 2021	JLG	Add V-I behavior plot, clarify current-limit behavior, correct getPortState bitfield