Overview

The USBHub3+ is an 8-port software-programmable USB 3.1 (Gen1; 5 Gbps) 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 USBHub3+ is established and maintained over the selected one of the two available host-facing ports or the dedicated Control Port connection.

The USBHub3+ 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 8 downstream ports
- Data lines, SuperSpeed lines 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 4A)
- Dedicated Control Port for software control independent of the selected host port
- Automatic or programmed selection for either of 2 host port connections
- All ports support USB link speeds up to 5Gbps
- Detect established link speed on each port:
  SuperSpeed (5Gbps)
  High Speed (480Mbps)
- Selectively enable USB charging mode behaviors:
  SDP (Standard Downstream Port) or
  CDP (Charging Downsteam Port) modes[1]
- Deliver up to 4.0A per port (in CDP mode)
- Set enumeration delay for discovery of attached downstream devices
- Boost USB 2.0 upstream and downstream signal levels
- DIN-rail mountable
- Alternate Eurostyle terminal block power input connector
- Certified to withstand +/- 15kV ESD strikes (IEC61000-4-2 level 4)

Description

The USBHub3+ gives engineers advanced flexibility and configurability over USB ports in testing and development applications.

The USBHub3+ hub architecture consists of two layers of internal hubs to achieve the 8-port hub function.

Each downstream USB channel implements separately and independently switched data lines and current-limited power lines. USB power, data and SS 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 USBHub3+ 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 USBHub3+. USBHub3+ 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}	0.0	36.0	V
Input Power		85	W
V_{bus} Ouput Power		65	W
Voltage on any V_{bus} line, upstream and downstream	0.0	5.1	V
Voltage on any USB D+/D-, upstream and downstream	-0.3	5.1	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, V_{supply}		9.0	12.0	24.0	V
Voltage on V_{bus} inputs and outputs	Hub un-powered	0	0	5.1	
	Hub powered; Port power enabled	4.5	5.0	5.1	V
	Hub powered; Port power disabled[2]	0.0	0.0	5.1	V

Table 3: Recommended Operating Ratings

[2] Applying voltage to downstream Vbus when hub is powered and port power is disabled for extended periods (>5 seconds) may cause damage to the port.

Block Diagram

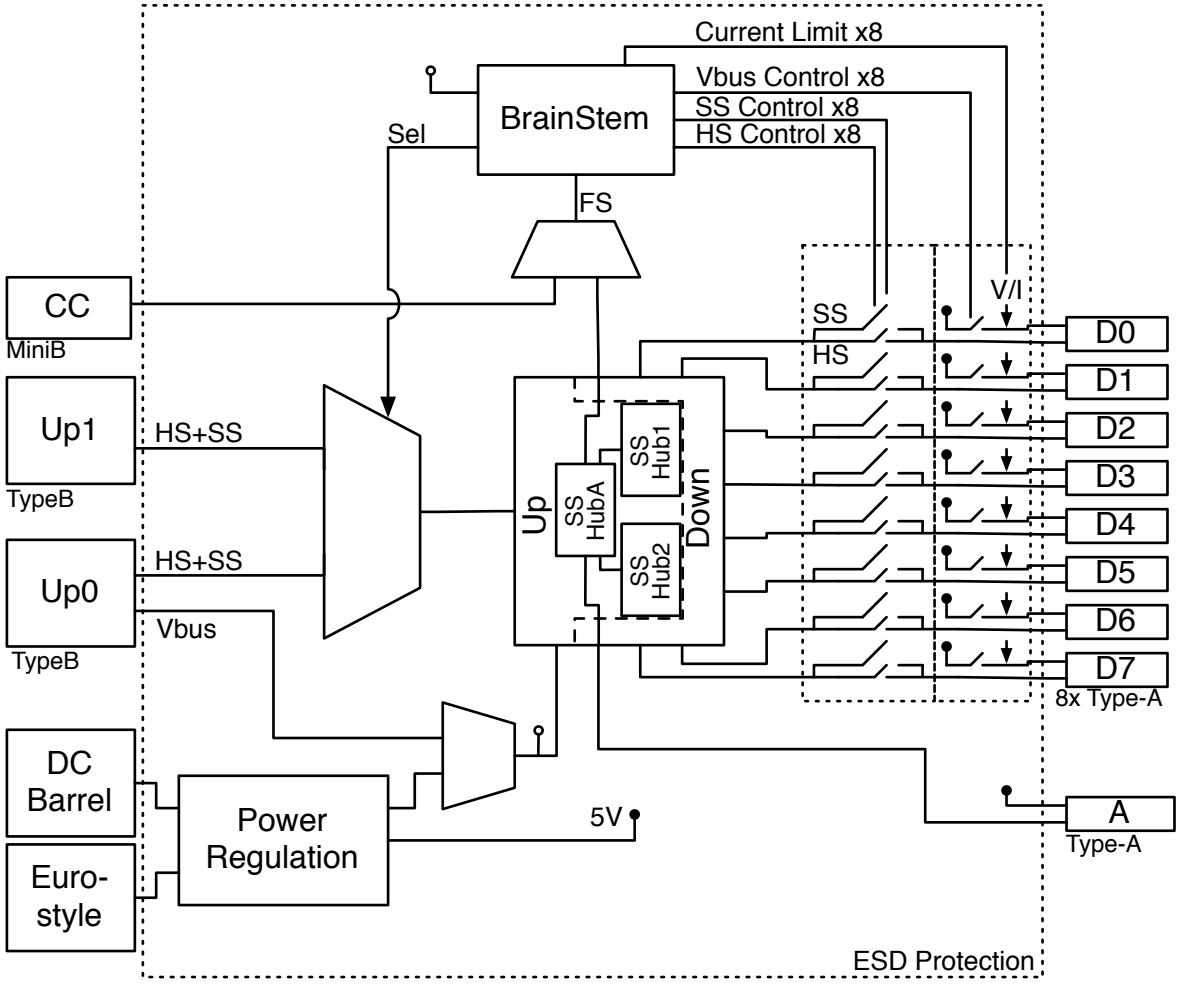


Figure 1: USBHub3+ Block Diagram

Typical Performance Characteristics

Values presented apply to the full operating temperature range.

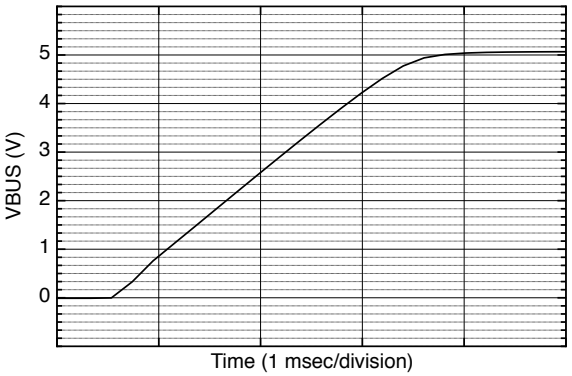
Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Input Power, W_{supply}, no downstream devices attached		-	2.2	-	W
V_{supply} Under Voltage Lockout (UVLO)		7.5	8.0	8.2	V
V_{supply} Over Voltage Lockout (OVLO)		26.0	26.8	27.8	V
Wide Input Range System Efficiency	@12.0V input, nominal 8A load[3]	84	-	92	%
USB Downstream Output Voltage, V_{bus}	No load on downstream USB ports	4.947	5.10	5.25	V
V_{bus} Measurement Resolution		-	8.0	-	mV
V_{bus} Measurement Accuracy		-2.0	-	2.0	%
V_{bus} Short-circuit Trip Current		4.8	5.0	5.4	A
V_{bus} Short-circuit Trip Time		-	0.7	-	µS
V_{bus} Short-circuit Average Current	After trip	0.3	0.5	2.0	A
V_{bus} Current Measurement Resolution		-	1.0	-	mA
V_{bus} Current Measurement Accuracy	V_{bus} current < 4.095A	-1.0	-	1.0	%
V_{bus} Current Measurement Range		0	-	4095	mA
V_{bus} Current Limit Trip Point Range	Software programmable	0	-	4095	mA
V_{bus} Current Limit Trip Point Resolution		-	1.0	-	mA
V_{bus} Overcurrent Trip Time	Time from overcurrent load to port power switch disconnect.	-	5.0	7.5	ms
USB SuperSpeed Data Rate	May depend on host or devices	-	-	5	Gbps
USB Hi-Speed Data Rate	May depend on host or devices			480	Mbps
V_{bus} Current Supply (SDP mode)	USB 2.0 data lines disabled or no USB host present, device limited	-	100	-	mA
V_{bus} Current Supply (SDP mode)	USB 2.0 data lines enabled and USB host present, device limited	-	500	-	mA

[3] Representative 8A load based on 8 USB downstream devices running in CDP mode consuming approximately 1.0A each.

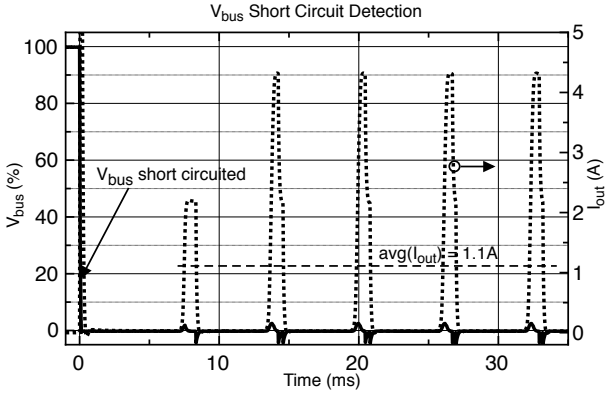
V_{bus} Current Supply (CDP mode)	USB 2.0 data lines enabled, USB host present, device limited	-	1500	-	mA
V_{bus} Current Supply (DCP mode)	USB 2.0 data lines enabled, no USB host present, device limited	-	5000	-	mA
Input current CH0 upstream port, I_{Vbus}	No V_{supply} present, USB 2.0 type-B cable	-	180	-	mA
Input current CH0 upstream port, I_{Vbus}	No V_{supply} present, USB 3.0 type-B cable	-	425	-	mA
Input current measurement resolution, I_{supply}	Through barrel jack or Euro-style connector only	-	4.0	-	mA
I_{supply} accuracy		-2.0	-	2.0	%
Input voltage measurement resolution, V_{supply}	Through barrel jack or Euro-style connector only	-	8.0	-	mV
Vsupply measurement accuracy		-2.0	-	2.0	%

Table 4: Typical Performance Characteristics

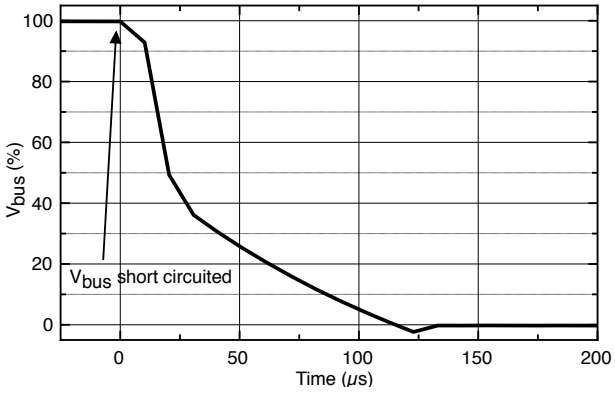
USB Downstream Enable into 100mA Load



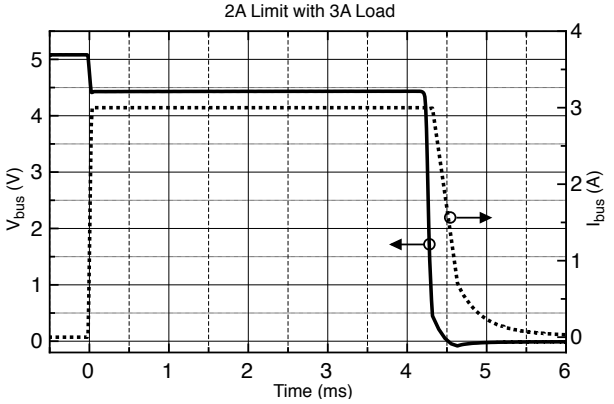
V_{bus} Short Circuit Detection



V_{bus} Short-Circuit Turn-off



V_{bus} Current Limit Trip
2A Limit with 3A Load



Device Drivers

The USBHub3+ leverages operating system user space interfaces that do not require custom drivers for operation on modern operating systems.

Outdated and unsupported operating systems 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. For example, Windows 7 requires the supplied INF to communicate with BrainStem USB devices.

Capabilities and Interfaces

The USBHub3+ 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 USBHub3+ are described in the following sections; refer to Table 10: Supported USBHub3+ 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.[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 USBHub3+ is assigned a unique serial number at the factory. This facilitates an arbitrary number of USBHub3+ 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 USBHub3+ 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 USBHub3+ 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. Upstream and downstream USB Boost settings, for example, 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 will return all settings to factory defaults: all ports' data (HS and SS) and power enabled, CDP mode, enumeration delay of 0, 4095mA 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)

USB Entity

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

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

USB Downstream Channels

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, SuperSpeed data, and V_{bus} lines simultaneously for a single port can be done by calling the following methods with channel in [0-7] being the port index:

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

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 [0-7]:

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

Manipulating just the USB 2.0 Hi-Speed data lines for a single port can be done by calling the following method with channel [0-7]:

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

Manipulating just the USB 3.1 SuperSpeed data lines for a single port can be done by calling the following method with channel [0-7]:

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

Manipulating just the USB V_{bus} line for a single port can be done by calling the following method with channel [0-7]:

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

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

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 [0-7], 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)
```

Current-limit trip point settings can be accessed for each port by calling the following methods with channel [0-7], where the second variable passed into the method is either the set value or the write location of the result:

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

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

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

with ch in [0-7] and speed values retuned as:

- 0 – N/A (device not enumerated)
- 1 – Hi-Speed device
- 2 – SuperSpeed device

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 have the hub signal to downstream devices they may consume upto 5A, the maximum allowed by BC1.2. If there is no upstream USB 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 USBHub3+. 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:

- 0 – Standard Downstream Port (SDP)
- 1 – Charging Downstream Port (CDP)

USB Downstream Enumeration Delay

Once a USB device is detected by the USBHub3+ 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, and is the time delay between enabling each successive downstream port from 0 to 7. 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 (SuperSpeed data and 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 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:

- 0 – normal drive strength
- 1 – 4% increased drive strength
- 2 – 8% increased drive strength
- 3 – 12% increased drive strength

USB Hub Upstream Channels

The USBHub3+ is perfect for environments where multiple devices need to be shared or switched between two host computers using two host (upstream) connections via USB standard-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:

- 0 – select upstream port 0 (`usbUpstreamModePort0`)
- 1 – select upstream port 1 (`usbUpstreamModePort1`)
- 2 – automatically detect upstream port (`usbUpstreamModeAuto`)
- 255 – disconnect both upstream ports (`usbUpstreamModeNone`)

Predefined C++ macros for these can be found in aProtocoldef.h, and Python's builtin help interface.

The default operational mode is to have the USB upstream be enabled through upstream port 1 and auto detect which USB port to use. Automatic detection uses the presence of V_{bus} on the USB type-B connector on upstream port 0. I.e. if a cable is connected in channel 0, then channel 0 will be used for upstream USB connection (see Figure 3).

BrainStem Control Port

The USBHub3+ also has a dedicated control channel on the USB mini-B. This is a full-speed USB 2.0 connection for BrainStrem interface only. No USB hub traffic can flow on this connection. When a cable is connected to the mini-B connector, the BrainStem device will be controlled only through this port. The USB 3.0 type-B connectors are then used only for USB hub traffic to connect downstream USB devices. When no cable is connected to the control port, BrainStem control is done through the selected upstream port.

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 Result 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	USB Channel 4 USB Hi-Speed Data Enabled
9	USB Channel 4 USB V_{bus} Enabled
10	USB Channel 5 USB Hi-Speed Data Enabled
11	USB Channel 5 USB V_{bus} Enabled

12	USB Channel 6 USB Hi-Speed Data Enabled
13	USB Channel 6 USB V_{bus} Enabled
14	USB Channel 7 USB Hi-Speed Data Enabled
15	USB Channel 7 USB V_{bus} Enabled
16	USB Channel 0 USB SuperSpeed Data Enabled
17	Reserved
18	USB Channel 1 USB SuperSpeed Data Enabled
19	Reserved
20	USB Channel 2 USB SuperSpeed Data Enabled
21	Reserved
22	USB Channel 3 USB SuperSpeed Data Enabled
23	Reserved
24	USB Channel 4 USB SuperSpeed Data Enabled
25	Reserved
26	USB Channel 5 USB Super Speed Data Enabled
27	Reserved
28	USB Channel 6 USB SuperSpeed Data Enabled
29	Reserved
30	USB Channel 7 USB SuperSpeed Data Enabled
31	Reserved

Table 5: Hub Operational Mode Result Bitwise Description

USB Hub State

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

```
stem.usb.getHubState(bank,status)
```

where *bank* can be [0-1], and the value *status* is 32-bit word, defined as the following:

Bit	Hub State (Bank = 0) Result Bitwise Description
0	USB CH0 device is attached
1:2	Reserved
3	USB CH0 error. See USB Hub Error Status Mapping
4	USB CH0 Hi-Speed Operation
5	USB CH0 SuperSpeed Operation
6:7	Reserved
8	USB CH1 device is attached
9:10	Reserved
11	USB CH1 error. See USB Hub Error Status Mapping
12	USB CH1 Speed Hi-Speed Operation
13	USB CH1 SuperSpeed Operation
14:15	Reserved
16	USB CH2 device is attached
17:18	Reserved
19	USB CH2 error. See USB Hub Error Status Mapping
20	USB CH2 Speed Hi-Speed Operation
21	USB CH2 SuperSpeed Operation
22:23	Reserved
24	USB CH3 device is attached
25:26	Reserved
27	USB CH3 error. See USB Hub Error Status Mapping
28	USB CH3 Hi-Speed Operation
29	USB CH3 Super Speed Operation
30:31	Reserved

Table 6: Hub State (Bank 0) Result Bitwise Description

Bit	Hub State (Bank = 1) Result Bitwise Description
0	USB CH4 device is attached
1:2	Reserved
3	USB CH4 error. See USB Hub Error Status Mapping
4	USB CH4 Hi-Speed Operation
5	USB CH4 SuperSpeed Operation
6:7	Reserved
8	USB CH5 device is attached

9:10	Reserved
11	USB CH5 error. See USB Hub Error Status Mapping
12	USB CH5 Speed Hi-Speed Operation
13	USB CH5 SuperSpeed Operation
14:15	Reserved
16	USB CH6 device is attached
17:18	Reserved
19	USB CH6 error. See USB Hub Error Status Mapping
20	USB CH6 Speed Hi-Speed Operation
21	USB CH6 SuperSpeed Operation
22:23	Reserved
24	USB CH7 device is attached
25:26	Reserved
27	USB CH7 error. See USB Hub Error Status Mapping
28	USB CH7 Hi-Speed Operation
29	USB CH37 SuperSpeed Operation
30:31	Reserved

Table 7: Hub State (Bank 1) Result Bitwise Description

USB Hub Error Status Mapping

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

```
stem.usb.getHubErrorStatus(bank,status)
```

where *bank* is [0-1] and the bits in *status* are defined in the following tables.

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

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

Calling this command clears the error bit flag (see Table 6-7) in the hub state as well as any errors indicated in the error state bit mask (Table 8-9).

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

Bit	Hub Error Status (Bank = 0) Result Bitwise Description
0	USB CH0 current limit exceeded
1:7	Reserved
8	USB CH1 current limit exceeded
9:15	Reserved
16	USB CH2 current limit exceeded
17:23	Reserved
24	USB CH3 current limit exceeded
25:31	Reserved

Table 8: Hub Error Status (Bank 0) Result Bitwise Description

Bit	Hub Error Status (Bank = 1) Result Bitwise Description
0	USB CH4 current limit exceeded
1:7	Reserved
8	USB CH5 current limit exceeded
9:15	Reserved
16	USB CH6 current limit exceeded
17:23	Reserved
24	USB CH7 current limit exceeded
25:31	Reserved

Table 9: Hub Error Status (Bank 1) Result Bitwise Description

USB System Temperature

The temperature of the USB subsystem in the USBHub3+ can be measured with:

```
stem.usb.getSystemTemperature(µC)
```

where temperature is in micro-degress Celcius.

USBHub3+ 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 USBHub3+ 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	
	getModuleBaseAddress	
	getModuleSoftwareOffset	
	setModuleSoftwareOffset	
	setHBInterval	
	getHBInterval	
	getRouterAddressSetting	
	getModule	
	getSerialNumber	
	getRouter	
	setRouter	
	getModel	
timer[0-8]	getExpiration	
	setExpiration	
	getMode	
	setMode	
usb[0]	setPortDisable	Channels 0-7
	setDataEnable	Channels 0-7

	setDataDisable	Channels 0-7
	setHiSpeedDataEnable	Channels 0-7
	setHiSpeedDataDisable	Channels 0-7
	setSuperSpeedDataEnable	Channels 0-7
	setSuperSpeedDataDisable	Channels 0-7
	setPowerEnable	Channels 0-7
	setPowerDisable	Channels 0-7
	getPortVoltage	Channels 0-7
	getPortCurrent	Channels 0-7
	getPortCurrentLimit	Channels 0-7
	setPortCurrentLimit	Channels 0-7
	setPortMode	Channels 0-7
	getPortMode	Channels 0-7
	getDownstreamDataSpeed	Channels 0-7
	getHubMode	
	setHubMode	
	getHubState	
	getHubErrorStatus	
	getSystemTemperature	
	getEnumerationDelay	
	setEnumerationDelay	
	clearPortErrorStatus	
	getUpstreamMode	
	setUpstreamMode	
	getUpstreamState	
	getUpstreamBoostMode	
	setUpstreamBoostMode	
	getDownstreamBoostMode	
	setDownstreamBoostMode	

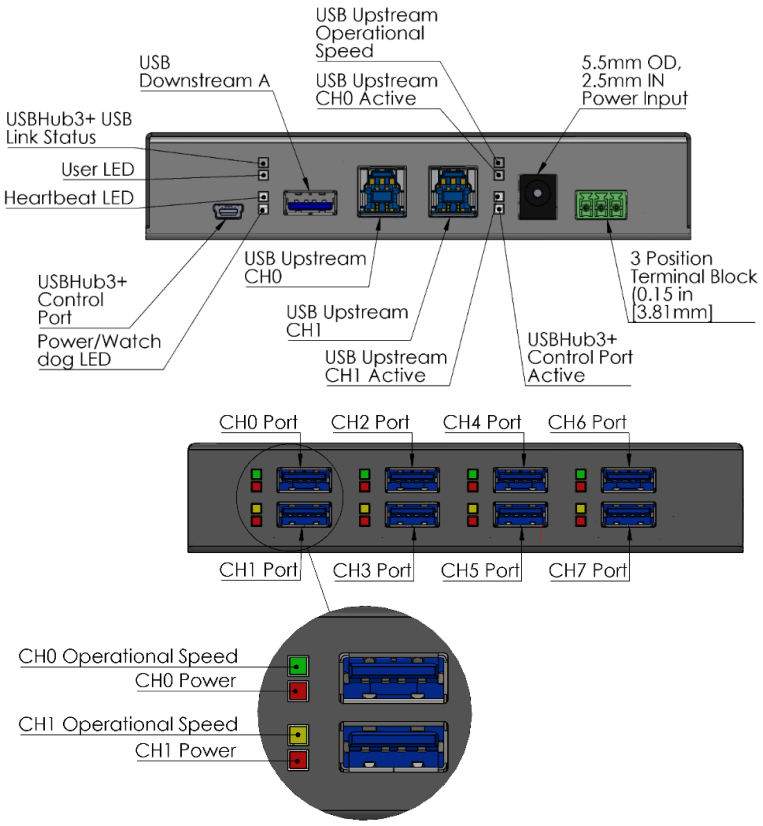
Table 10: Supported USBHub3+ BrainStem Entity API Methods[5]

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

LED Indicators

The front and back of the USBHub3+ 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
Link Status LED	Yellow	On once a host device has enumerated the BrainStem controller
User LED	Blue	Can be manipulated through any of the available APIs
Heartbeat LED	Green	Indicates active BrainStem connection; pulses at a rate determined by the system heartbeat rate
Power/Watchdog LED	Red and flashing blue	Solid red indicates the system is powered. Flashing blue is indication the internal watchdog is running and the USBHub3+ firmware is healthy
Upstream Operational Speed LED	Yellow or green	Upstream enumeration speed to host: green for SuperSpeed; yellow for Hi-Speed or lower USB 2.0 speeds.
Upstream 0 LED	Green	Indicates an active connection on upstream port
Upstream 1 LED	Green	
Control Port LED	Yellow	
Downstream Operational Speed LED	Yellow or green	Downstream device enumeration speed: green for SuperSpeed; yellow for Hi-Speed or lower USB 2.0 speeds; off when no device is enumerated
Downstream Power LED	Red	LED is on when downstream V_{bus} is enabled



Host Port Control Application Notes

The two upstream ports can be connected to two different host computers. Due to limitiations of USB specification and architecture, only one host computer can access downstream USB ports at a time. Through the BrainStem APIs, the upstream port used can be specifically selected, or the system can automatically select one. In addition, a dedicated USB control port is available that provides a constant connection to the BrainStem controller independent of which USB host port is connected. If the dedicated control port is not used, the BrainStem connection is shared with the active upstream connection. Using the control port provides the capacity to completely disconnect both USB upstream host connections. Figure 3 details the decision flow for selecting the port for BrainStem communications.

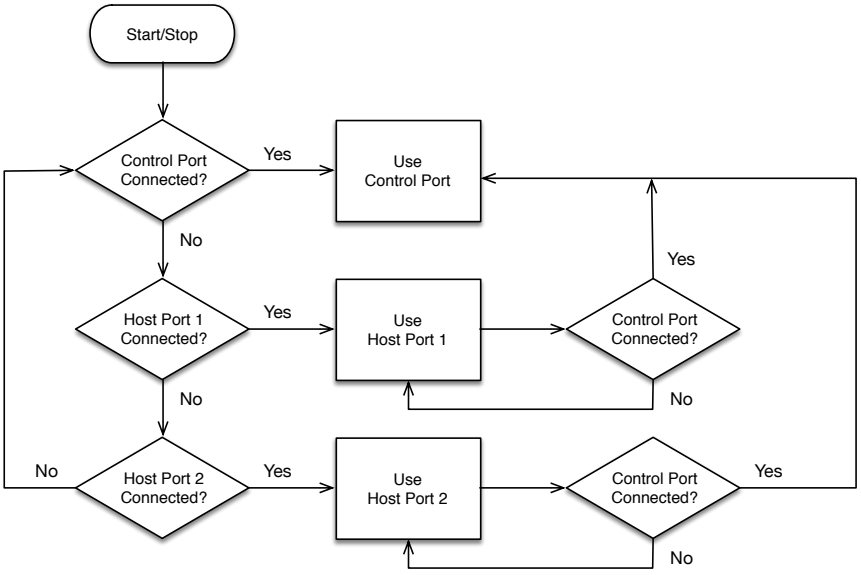


Figure 2: Host Port Control Flow

Mechanical

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

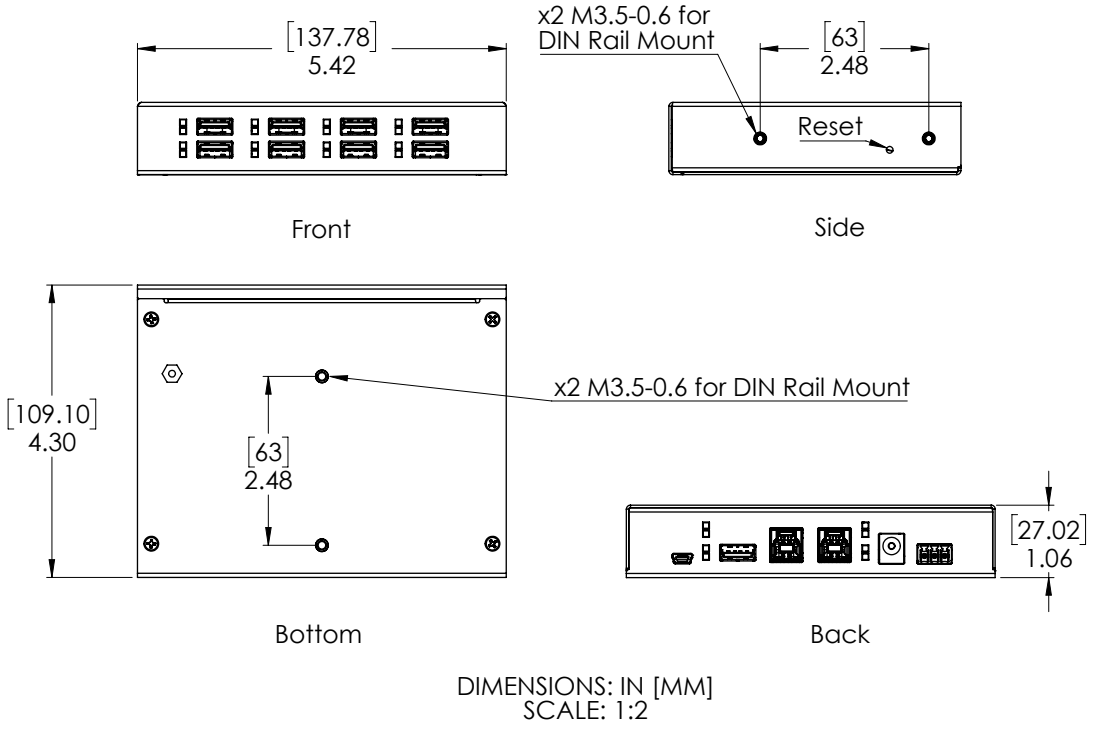


Figure 3: USBHub3+ Mechanical

DIN Rail Mounting

DIN rail mounts have been designed into the USBHub3+ case with an appropriate clip as often used for industrial control equipment. Mounting clip hardware is not included with the USBHub3+. The mounting holes are compatible with many widely available “small” DIN rail mounting clips, and Acroname part number C31-DINM-1. The USBHub3+ can be mounted in two positions:

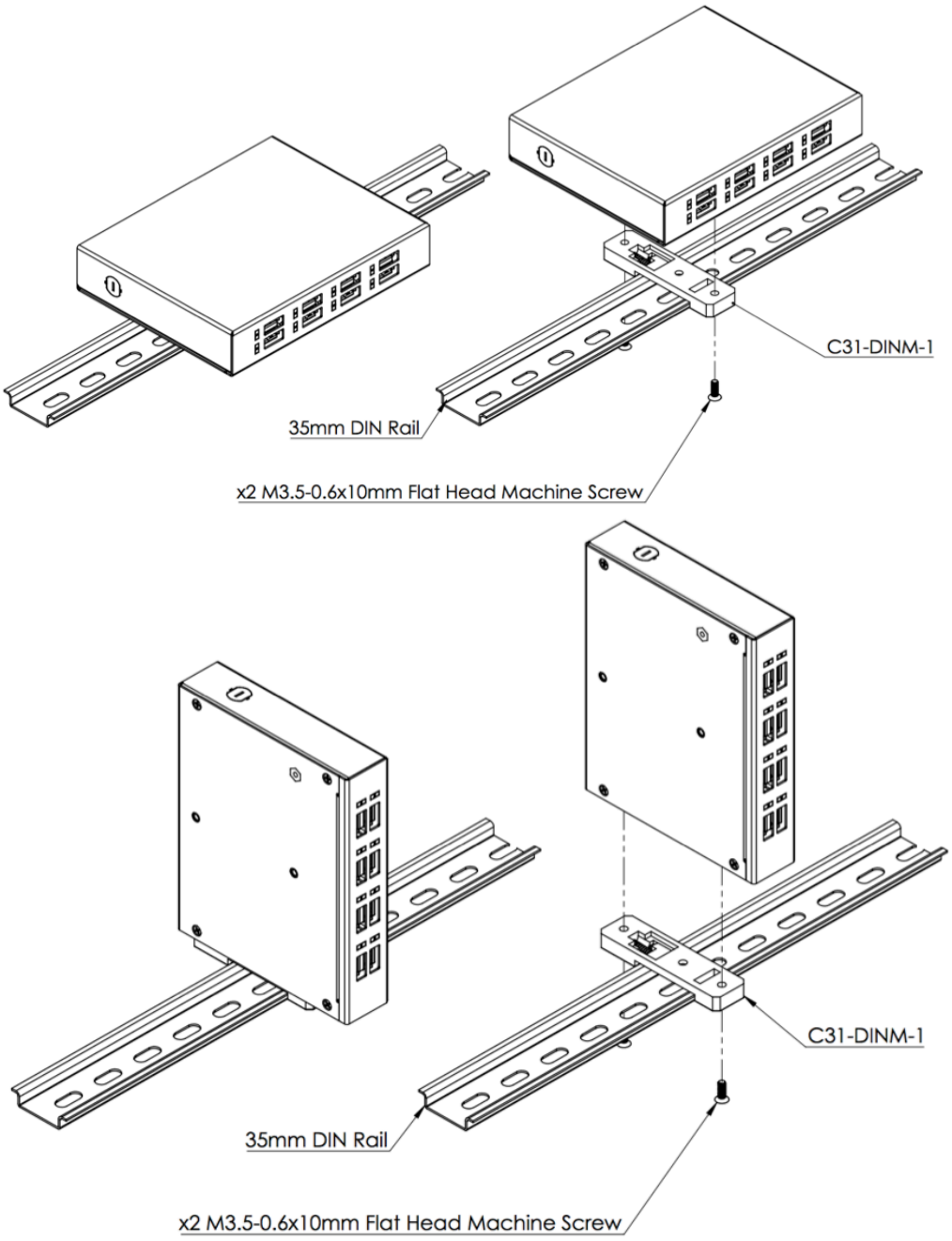


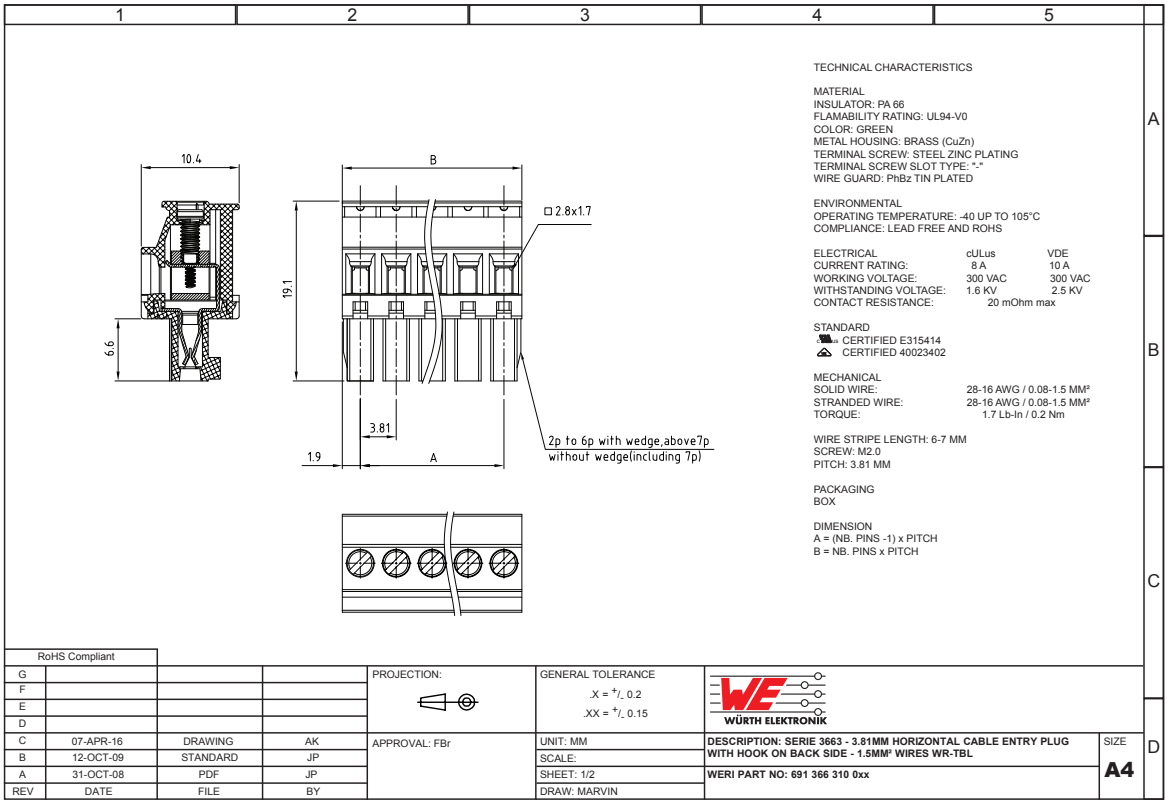
Figure 4: USBHub3+ DIN Rail Mount

Input Power Connections

The USBHub3+ can be powered in three ways: bus powered from upstream port 0; the DC “barrel-jack”; a “Euro-style” terminal block. When bus-powered from upstream port 0, the system’s downstream ports are automatically disabled and cannot be disabled. This power mode is useful for testing while developing code. To enable downstraem ports, external power must be supplied.

The DC barrel-jack is a standard 3.5mm outside diameter, 2.5

In addition to the standard barrel jack power input an alternative power input, a Euro-style 3-pin terminal block (“Euroblock”), is also provided. This connector provides the additional benefits of a dedicated earth connection and a higher current rating (8 A) than the barrel connector (5 A). Many manufacturers make compatible connectors; one example is the Wurth Elektronik Serie 3663- 3.81mm Horizontal with Hook on Back Side WR-TBL, part number 691366310003 (see diagram below).



Source: http://katalog.we-online.com/en/em/TBL_3_81_3663_HORIZONTAL_W_HOOK_ON_BACK_SIDE_69136631000X

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
1.0	September 2016	JTD	Initial Release
1.1	September 2016	JLG	Update block diagram
1.2	October 2016	LCD	Update Overview, Features, Description Text
1.3	October 2016	LCD	Update ESD compliance info
1.4	January 2017	JLG	Add Vbus overvoltage information; add voltage and current measurement accuracy
1.5	February 2017	JLG	Add port state to saved parameters