Thank you for your purchase of **ControllerHub 8** from Amelia's Compass! This manual will help you get the most out of it.

Introduction – The Problem:

Electronic audio processors and instruments are extremely powerful, whether they take the form of hardware or software. A large number of user-adjustable parameters are usually provided so that the sound can be precisely controlled. But a lot of this power remains untapped in live performance, because there's not been an easy way to control the user-adjustable parameters "in the heat of battle".

This is generally not as much of a problem when we're "off-line" - sitting on the couch with the iPad or laptop, or recording in the basement studio. In these situations we have all the time we need to navigate through "menu hell", drilling down to whatever parameter we'd like to adjust, and then manipulating it using either a mouse or the touchscreen.

But menu hell is completely unworkable in live performance! We can't be manipulating a mouse and concentrating on a laptop screen, while simultaneously trying to play an instrument. Here we need real, physical manipulators like faders & knobs instead of pull-down menus and touchpads.

Even hand-operated physical hardware is not a viable solution if both hands are occupied while playing an instrument. We need to use our feet! We need movable footpedals and stomp-able footswitches, wired somehow to the audio parameters we need to control.

How does ControllerHub 8 solve the problem?

ControllerHub 8 is a connectivity hub that enables a modular control strategy for your computer-based, live performance audio. Use your favorite expression pedals and footswitches for computer parameter control!

On one side, ControllerHub 8 has eight ¼" TRS inputs for connection of your pedals and buttons:

On the other side, it has both MIDI and USB I/O for connection to the rest of your audio system:

So you can use any number of your favorite expression pedals…

…plus any number of your favorite on/off buttons…

…and connect them all, in any combination, to create your own custom control setup:

A single ControllerHub 8 provides eight controller inputs. You decide which of them will be used to connect expression pedals, and which will be used to connect on/off buttons. For larger systems, ControllerHub 8 can be used in multiples to create ever more complex modular control setups.

External Connections

On one end of ControllerHub 8 there are eight ¼"TRS input jacks:

Each of these jacks enables the connection of either a variable footpedal (such as an expression pedal or some types of volume pedals) or a stomp-able footswitch (such as an on/off pushbutton, or a keyboard sustain pedal). It was a bit of an engineering challenge to allow the inputs to switch-hit between pedals and buttons, but it makes things so much easier.

If you connect a variable expression pedal to a ControllerHub 8 input, then you must use a cable with TRS connectors at each end (in other words, NOT a normal instrument cable). This is because expression pedals themselves have a 3-conductor TRS jack, so a TRS jack is needed on both ends of the cable, and of course the cable itself must be TRS. On the other hand, if you connect an on/off footswitch to a ControllerHub input, then you can use either a ¼" TRS cable (3 conductor), or a normal ¼" instrument cable (2 conductor).

Some volume pedals seem to work OK with ControllerHub 8 (but not all!). If you wish to try using a volume pedal, connect it with a standard ¼" instrument cable to the volume pedal's output jack. Volume pedals are not ideal since they typically use an audio-taper potentiometer, whereas expression pedals use linear-taper pots. A linear-taper pot is generally better suited for MIDI parameter manipulation. Also, volume pedals which are not potentiometer-based may not work with ControllerHub 8 at all.

On the other end of the ControllerHub 8 there are USB and MIDI connectors:

This is how you connect to the rest of your system using either USB or MIDI, whichever you're set up for. You can even use both at the same time, and this opens up some interesting possibilities which we'll get into later. Take a look at the **Combining Multiple ControllerHub 8 Units** section to find out more.

Here's a simple **Keyboard Setup**:

The USB keyboard and ControllerHub 8 are both connected to the laptop or tablet via USB[1]. The expression pedals connect to the ControllerHub 8 with ¼" TRS cables and the buttons with standard ¼" instrument cables.

In this example we assume a group of four controllers for the keyboard – two expression pedals, a sustain pedal, and a general-purpose footswitch. The physical controllers are mapped to desired destination parameters in whatever audio app is running on the computer. For example it seems reasonable that one of the expression pedals would be mapped to the keyboard's volume parameter, while the sustain pedal would presumably be mapped to the sustain parameter, etc. Perhaps the button turns on a Leslie effect for organ plugins, and so on.

It's interesting to note that ControllerHub 8 has no knowledge about these mappings. Its job is simply to report changes of the controller positions to the laptop or tablet; it doesn't take part in the subsequent mapping of controller message streams to the various application parameters. Each application will have its own method of making these controller assignments. We'll try to post videos of using ControllerHub 8 with a number of popular applications, but of course the various applications' manuals will always be the ultimate source for up-to-date information.

[1] If the laptop or tablet has fewer than the required number of USB ports, a USB Hub can be inserted between the USB devices and the laptop or tablet.

Here's a typical **Guitar Setup**:

The guitar's audio signal is applied to one of the inputs of a typical USB audio interface, and the amp is driven by one of the USB audio interface's outputs. Both the audio interface and the ControllerHub 8 then connect to the computer via USB.

This example shows just four controllers connected to ControllerHub 8, but of course you can connect as many as eight.

Internal MIDI Pathways

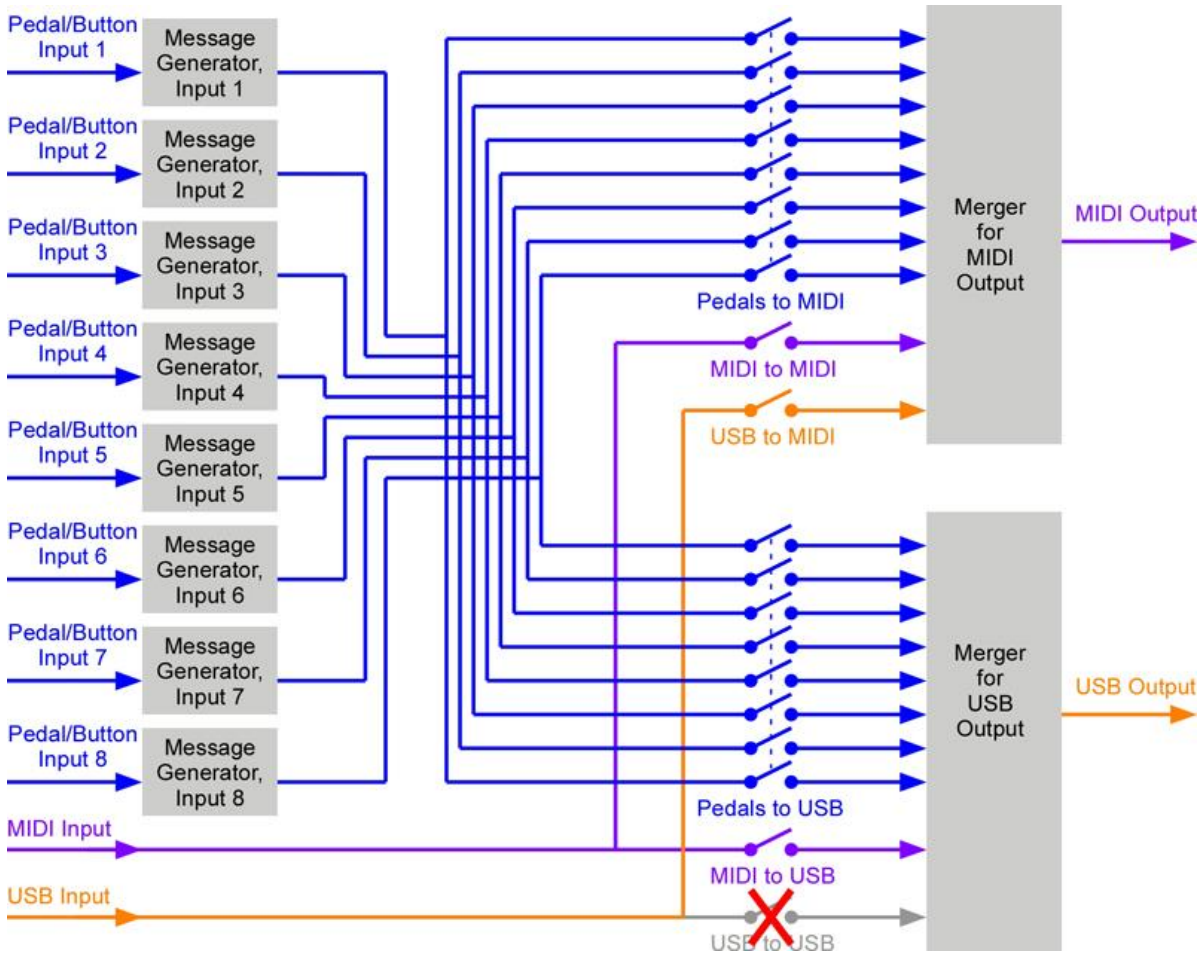
MIDI Sources:

ControllerHub 8 has eight ¼" TRS pedal/button inputs that cause the generation of MIDI messages to the rest of the audio system. MIDI can also be driven into the ControllerHub 8 from two external sources; these are the USB and MIDI In connections. So inside ControllerHub there are ten potential MIDI sources, as shown coming in on the left side of the diagram below.

MIDI Destinations:

ControllerHub 8 also has two connections from which MIDI emerges to be sent to other system elements; these are the USB and MIDI Out connections. The MIDI destinations are shown on the right of the diagram.

Inside ControllerHub 8, there is a MIDI patchbay which merges all ten MIDI sources to both MIDI destinations. Well, I take that back – we don't allow messages coming in on USB to be echoed back. But with that one exception, all sources are merged to all destinations.



Inputs are connected to outputs by means of five switches, as shown in the above diagram.

These switches are called:

- Pedals to MIDI (eight switches ganged together)
- MIDI to MIDI (i.e., MIDI Thru)
- USB to MIDI
- Pedals to USB (eight switches ganged together)
- MIDI to USB

The factory default has all of these switches set to ON, implementing an "All Merged to All" connectivity. But you can edit the patch points to disable any of them, either singly or in arbitrary combinations.

LED Indicators

ControllerHub 8 is well provisioned with LEDs that may be very helpful for cable troubleshooting or confidence monitoring.

Input LEDs

On the input side of the ControllerHub 8, there is a group of three LEDs on the lower right, and each input jack also has its own yellow LED.

- Of the three LEDs in the lower right-hand group, the green LED isn't labelled. Its function is simply to indicate that the unit is powered. It blinks softly so you know there is something going on inside.

- The yellow "Rx" LED blinks whenever a message is received via MIDI or USB inputs. It also blinks when an editing message is received from an external source, to change an internal parameter.

- The "Tx" LED blinks whenever the unit sends a message via MIDI or USB.

Quite often both the Tx and Rx LEDs will blink together. This means that an input message has been received that has also been passed through to one of the outputs.

The last function of this group of three LEDs is to flash in sequence at power-up and after a write to flash.

Each of the ¼" TRS connectors also has its own LED. If nothing is plugged into a particular input connector, its LED will be dark. As soon as something is plugged in, however, the input's LED is softly illuminated. Finally, when the connected device is manipulated in such a way that a MIDI or USB message needs to be sent, then the input's LED will flash more brightly.

USB / MIDI LEDs

On the MIDI/USB side of the ControllerHub 8, there are five LEDs:

- The green one next to the wall wart connector is a duplicate of the other green one on the ControllerHub 8's input end; its function is simply to indicate that the unit is powered, and it blinks softly to let you know there is something going on inside.

- There are two yellow LEDs next to the USB MIDI connector, labelled "Rx" and "Tx". When messages arrive via USB from some external source, the "Rx" light briefly illuminates. When the ControllerHub 8 generates a message and sends it to the rest of the system via USB, the "Tx" LED briefly illuminates.

- There is a yellow LED next to each of the MIDI connectors. When messages arrive via MIDI from some external source, the LED next to the MIDI IN connector briefly illuminates. When the ControllerHub 8 generates a message and sends it to the rest of the system via MIDI, the LED next to the MIDI OUT connector briefly illuminates. In addition, if the ControllerHub 8's Power-Over-MIDI function is enabled, the LED next to the MIDI IN connector is softly illuminated. In this case, incoming messages will still cause this LED to flash more brightly.

The final purpose of the LEDs is to indicate when an internal parameter editing session has been saved to the unit's internal non-volatile flash storage. You can edit as many parameters as you like, and whenever there's a 10-second gap in your editing, the unit saves to flash. During the write, the unit does a chase-light pattern on the LEDs.

Combining Multiple ControllerHub 8 Units

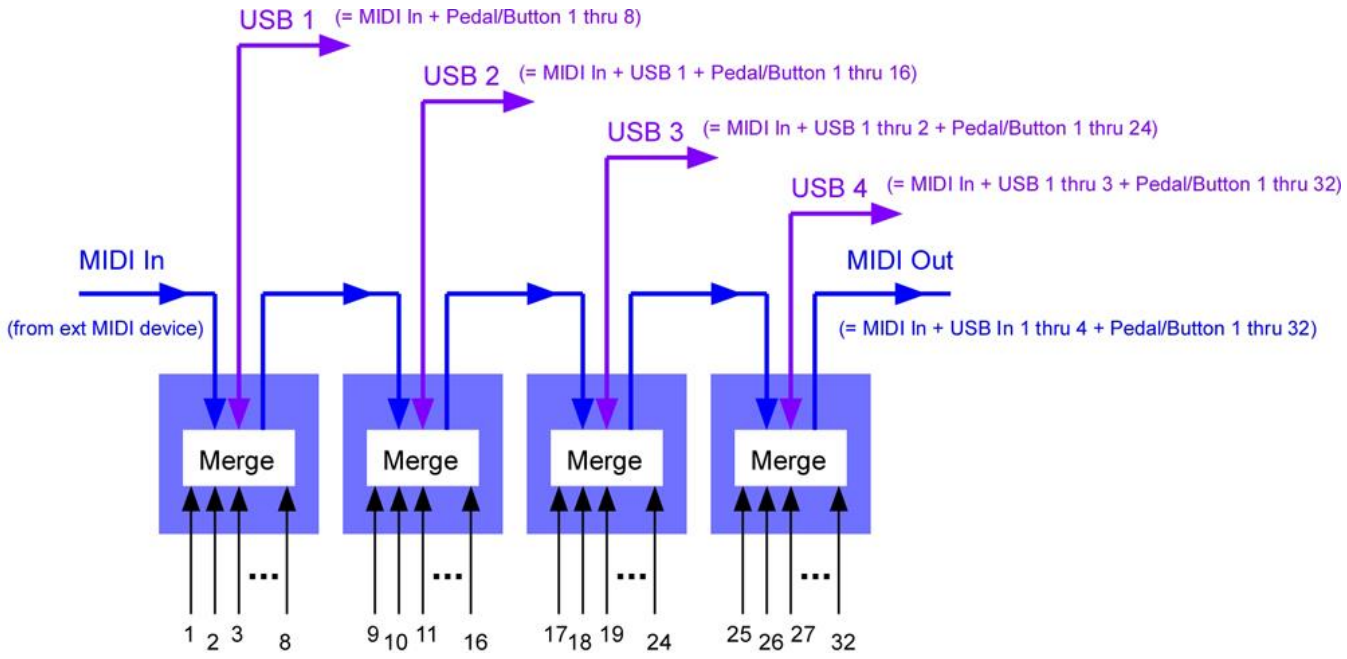
There may be times when more than eight physical controllers are desirable. In such cases, you can simply use more than one ControllerHub 8 in a combined setup.

There are several possibilities for combining multiple ControllerHub units to create larger systems with greater numbers of Pedals, Buttons, and other connected MIDI and USB devices. The combining methods are daisy-chaining, cross-connecting, looping, and USB hub.

Daisy-Chaining:

Using the MIDI connections, you can daisy-chain multiple units by connecting the MIDI Output of one unit to the MIDI Input of the next unit. The output of the last unit in the chain finally goes to the rest of the system to be controlled.

Here's a block diagram of what happens when you daisy-chain multiple ControllerHub 8 units (in this example, four ControllerHub 8 units are daisy-chained):

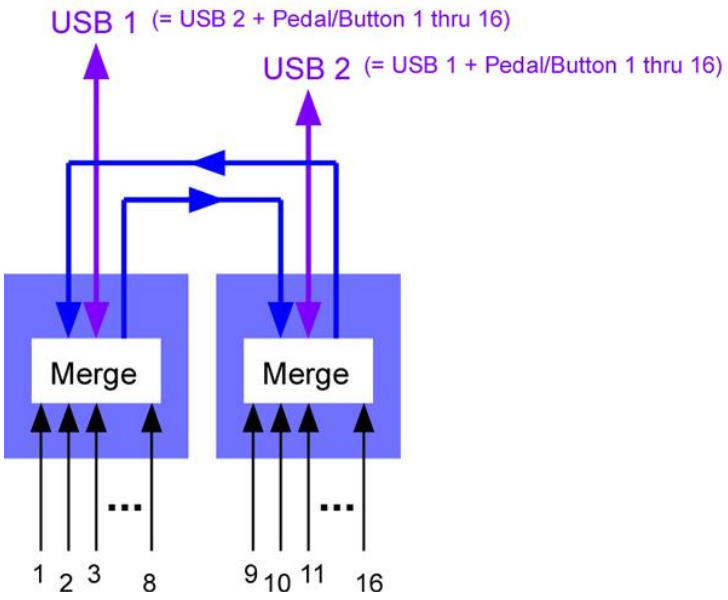


Notice that as you move to the right, more and more things have been merged together to create each output.

Note: Maybe this is obvious, but it might be worth pointing out that not all of the USB or final MIDI Out connections need be utilized. For example, perhaps you'll only use the last USB connection on the right, or maybe only the final MIDI Out connection. After all, the USB connections further to the left are less useful, because they include the merger of fewer and fewer inputs as you go left. Then again, maybe someone out there will find them useful!

Cross-Connecting Two Units:

By cross-connecting the MIDI Input and Output jacks of two units, you can create a bi-directional connection between two USB devices:



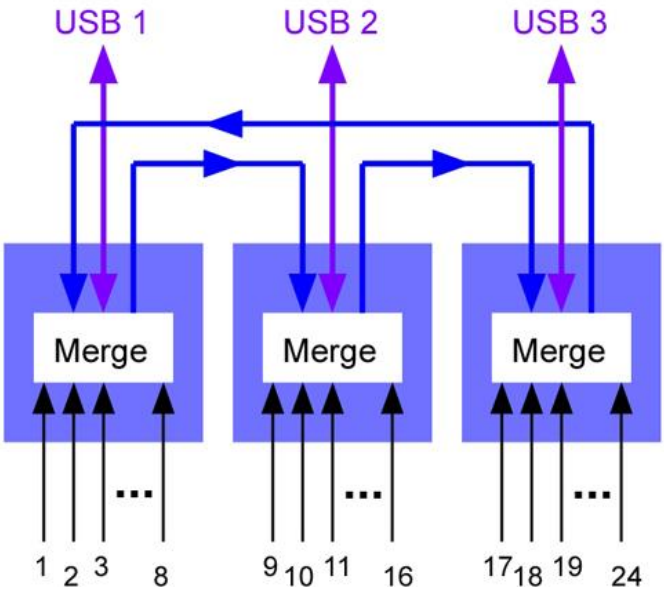
Since all the MIDI connections are used up in the connection of the loop, only USB connections remain to connect external devices.

For example, this configuration could give you a wired connection between an iPad and a laptop.

Important: When cross-connecting, make sure to disable the internal MIDI-to-MIDI switch on at least one of the units, otherwise messages entering the loop will circulate and repeat forever!

Looping:

Looping is like a hybrid between cross-connecting and daisy-chaining, where you use the MIDI connections to create a loop out of more than two units.



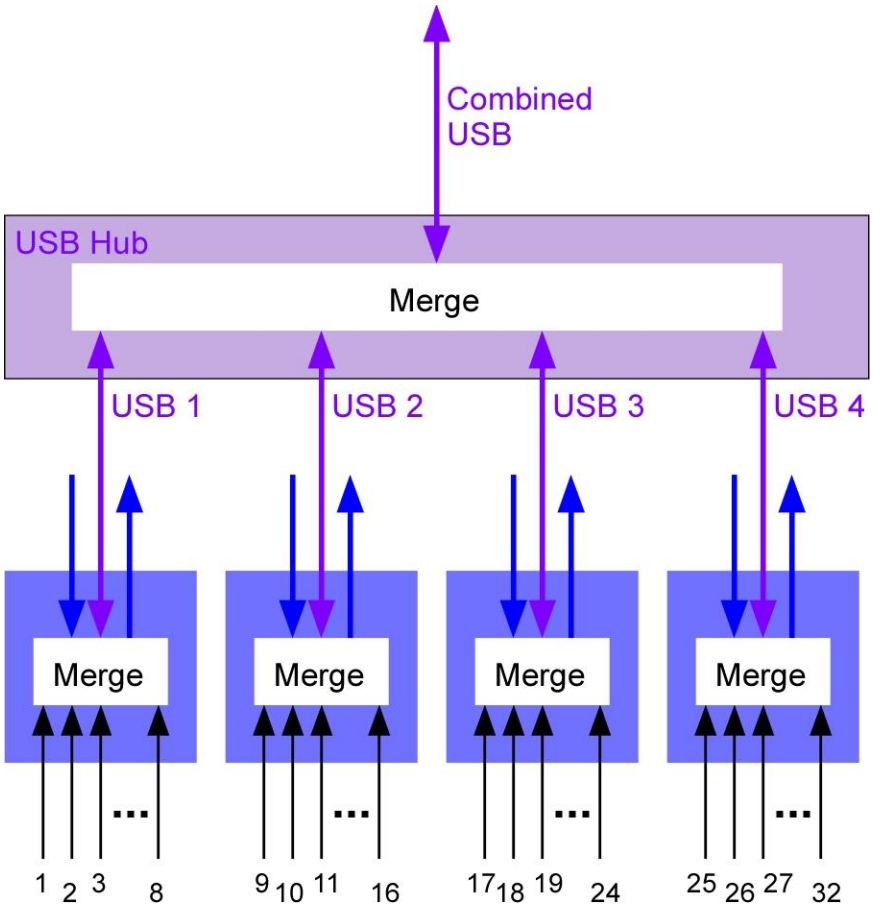
Since all the MIDI connections are used up in the connection of the loop, only USB connections remain to connect external devices.

Here, too, it is very **important** that at least one of the units disables the MIDI-to-MIDI switch, to prevent MIDI messages from circulating forever around the loop!

Note: Make sure to read the Power section, because the Power-over-MIDI option allows you to **eliminate wall warts** when combining units using MIDI cables!

USB Hub (or connection to multiple laptop USB ports):

This is fairly obvious; just connect all the ControllerHub 8's to a USB Hub, and then connect the upstream USB from the hub to the rest of the system. Or simply connect each ControllerHub 8 to a different USB port if your computer has more than one.



USB Hubs are pretty small and fairly cheap, so there's not much of a space or cost disadvantage to this approach. On the other hand, an important advantage is that all ControllerHub 8's MIDI ports are available for connection to other devices.

When combining multiple ControllerHub 8's using a USB hub (or when using multiple laptop USB ports), it's important to set each ControlHub 8's Unit ID to a unique number, or the USB Host will not be able to tell them apart. In all of the previous multi-unit connection schemes, the USB host can only "see" one of the units, so Unit ID's are optional. But with a USB Hub or multiport laptop connection, the host requires each ControllerHub 8 to be set to a unique Unit ID. In the parameter editing section of this manual you can find out all you need to know about setting Unit ID's.

Powering ControllerHub 8 (either singly or in groups)

Powering a single ControllerHub 8 - a single ControllerHub has two ways of obtaining power:

- Wall Wart
- USB

If you have just one ControllerHub 8, then it can't obtain power via Power-Over-MIDI, because only another ControllerHub 8 can be the source of that power.

It's OK to hook USB up to a ControllerHub 8 that is already being powered by a Wall Wart, or to hook a wall wart up to a unit that's already being powered via USB.

Powering a ControllerHub 8 that is one of multiple ControllerHub 8's in a group - a ControllerHub 8 that is part of a multi-unit group has three ways of obtaining power:

- Wall Wart
- USB
- Power-Over-MIDI (PMIDI)

Just like the case of a single ControllerHub 8, a ControllerHub 8 that is at the head of a daisy chain can't obtain power via Power-Over-MIDI, because only another ControllerHub can be the source of that power. And it's probably fairly obvious that in any group of ControllerHub 8's, at least one of them must obtain power from something other than PMIDI. In a daisy-chain, the ControllerHub 8 at the head of the chain must be the externally-powered unit. In a loop of ControllerHub 8's, it doesn't really matter which one obtains power from outside the loop, just as long as one of them does.

We recommend limiting daisy-chain length to four when using the wall wart available from Amelia's Compass. If you want to get a bigger wall wart, you could possibly daisy-chain more of them together, but eventually one of two things will limit the length of the chain. Either the wall wart will not be able to supply enough current and the voltage will drop below the minimum required, or the small voltage drops incurred in each length of the chain will result in MIDI outputs of ControllerHubs at the end of the chain not being recognized, because they don't have the full 5V to work with that the MIDI spec requires.

You don't have to worry about activating Power-Over-MIDI to a ControllerHub 8 that is already being powered by other means. But you should be careful NOT to activate Power-Over-MIDI from a ControllerHub that is receiving MIDI from a device other than another ControllerHub. In other words, send PMIDI to other ControllerHubs ONLY.

Finally, PMIDI only works when there is a fully-wired, 5-conductor MIDI cable attached between the source and destination. Generally speaking, I think it's safest to assume that if a MIDI cable manufacturer does not specifically claim that the cable in question has all five pins wired, then it does NOT. But a number of manufacturers offer 5-pin MIDI cables and they don't look to be any more expensive than cables not claiming to be 5-pin.

Factory Default Internal Parameter Settings

When a controller device connected to a ControllerHub 8 input is moved, what MIDI message is generated? There are two possible answers to this question. First, ControllerHub 8 is pre-configured with factory default settings. The default settings ensure that each input sends a unique message stream, so that the destination software can easily differentiate between them. Alternatively, you can change the default settings so that the message types of your choosing are sent by each of the inputs. After these changes, you can always reset the unit back to its factory defaults, if desired.

Factory Defaults:

Input 1 sends MIDI CC#7 (volume) messages on MIDI channel 1
Input 2 sends MIDI CC#7 (volume) messages on MIDI channel 2
Input 3 sends MIDI CC#7 (volume) messages on MIDI channel 3
Input 4 sends MIDI CC#7 (volume) messages on MIDI channel 4

Input 5 sends MIDI CC#64 (sustain) messages on MIDI channel 1
Input 6 sends MIDI CC#64 (sustain) messages on MIDI channel 2
Input 7 sends MIDI CC#64 (sustain) messages on MIDI channel 3
Input 8 sends MIDI CC#64 (sustain) messages on MIDI channel 4

The first four inputs have their Button Mode function disabled, meaning they are all set up to expect variable expression pedals, not two-state buttons. The second set of four inputs have their Button Mode function enabled, meaning they are set up to expect two-state buttons, not pedals.

Polarity is positive for the pedals and inverted for the buttons.

Toggle is OFF (though it is ignored in any case for the pedal inputs).

The default MIDI patchbay settings are all active, meaning all inputs are merged to both outputs.

Power-Over-MIDI is deactivated (i.e., MIDI phantom power).

Auto Repeat is deactivated for all inputs.

Finally, the Unit ID is set to zero, meaning that effectively there is no Unit ID. This prevents the case where multiple units have duplicate ID's (until the ID's can be set to unique numbers by the user).

This default configuration will work for the majority of people that use ControllerHub 8 to interact with software programs, because generally the ultimate target parameter is completely map-able within that software. For example, MainStage even has an "Assign" function that maps whatever message comes in next to the parameter being mapped. So you hit Assign, move a pedal, and that pedal is now mapped to that parameter. The actual message sent by the pedal is irrelevant; it is only necessary that each controller input sends a unique message type. The factory default ensures that this is the case.

On the other hand, you may need to talk to a hardware device whose MIDI channel is set to something other than 1 through 4, for example. In order to talk to it, you will have to edit the MIDI channel being used in the messages sent by ControllerHub 8.

Or you may wish to activate MIDI phantom power, so that only one of your units needs to be hooked up to power. The factory default has this deactivated.

In cases such as these, you'll need to edit the ControllerHub 8's internal parameters to something other than their factory default settings.

The next section describes the process for editing ControllerHub 8's internal configuration parameters.

Editing ControllerHub's Internal Configuration Parameters

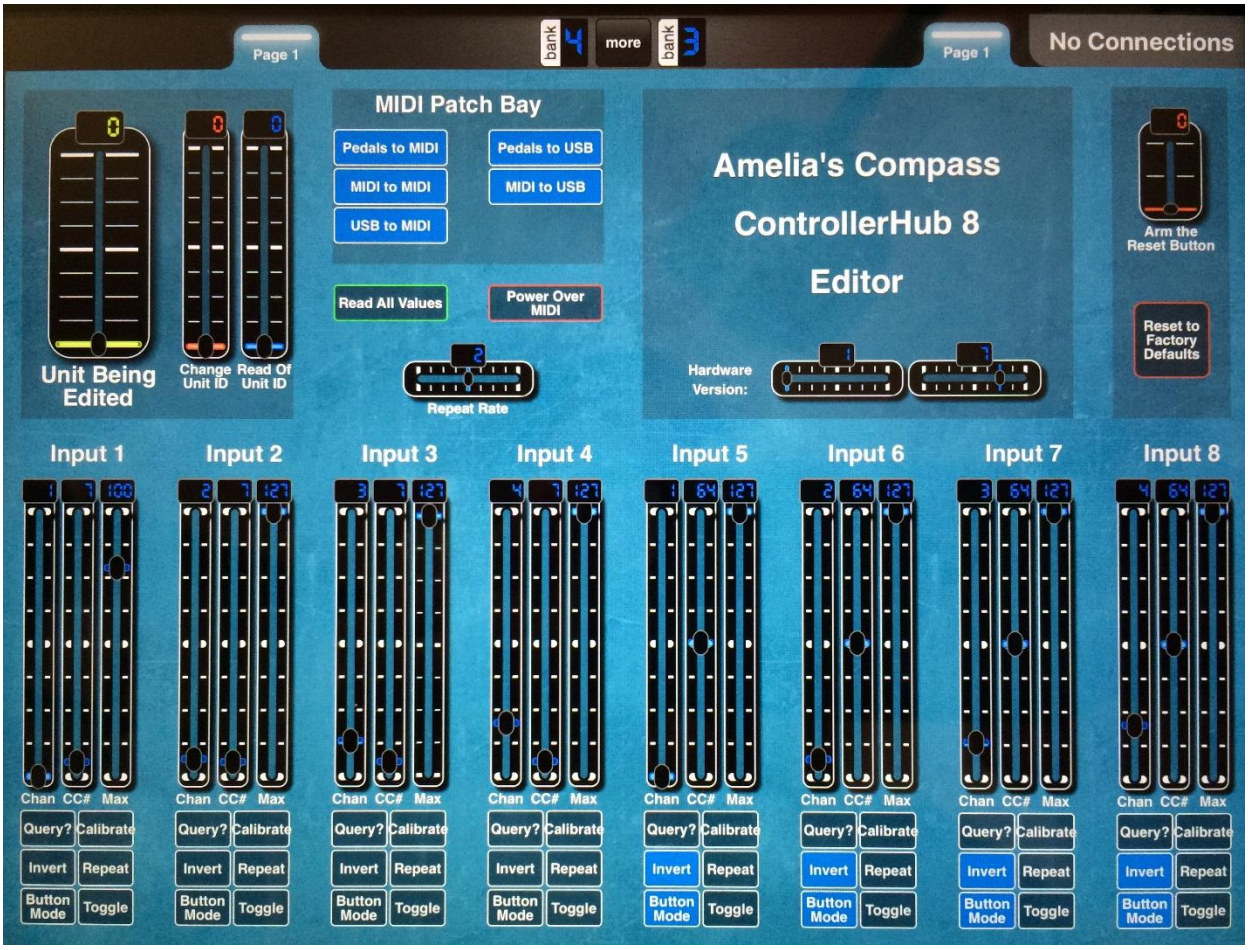
All of ControllerHub 8's internal configuration parameters are editable via MIDI System Exclusive messages (SysEx). So it's possible to edit ControllerHub 8's internal configuration parameters with anything capable of sending user-created SysEx messages. Later on we'll show you a chart of the MIDI SysEx implementation to help you do this.

But first, let's describe the easiest way to edit ControllerHub 8 parameters, and that is to use MIDI Designer software.

MIDI Designer is an IOS app sold by Confusion Studios, LLC. It runs on an iPad, and is available for download from the App Store. Using an iPad, MIDI Designer, and Apple's Camera Connection Kit (or IK's iRig MIDI 2), it's very easy to customize the operation of ControllerHub.

MIDI Designer is a build-it-yourself editor for all things MIDI. It gives you the ability to design your own editor/librarian for whatever MIDI equipment you own. By dropping knobs, sliders, and buttons on the screen, you can create your own graphic user interface screen. These screens are called "templates" in MIDI Designer parlance.

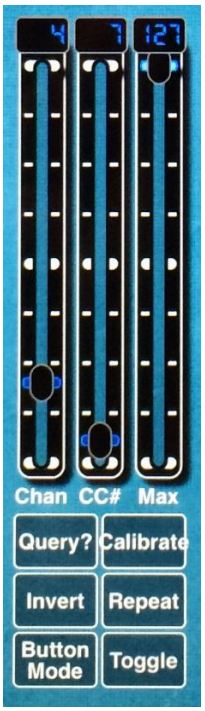
Using MIDI Designer, we've created an editor/librarian template for ControllerHub 8:



Refer to Appendix A for directions on how to download & install MIDI Designer, and how to obtain the ControllerHub 8 editing template.

Notice that there are eight main configuration strips, one for each ¼" TRS Pedal/Button input. In addition, there are the five MIDI Patch Bay switches, plus a few other controls.

Pedal/Button Input Parameters:



Each Pedal/Button Input has seven user-editable parameters. These allow the user to change the details of the message stream created by ControllerHub when the physical controller connected to the input is seen to change.

While these messages are restricted to the MIDI Continuous Control message type, there is a lot of variability available with this message type. Since it is assumed that the messages are routable in the destination software, it is really only necessary that the Pedal/Button inputs send messages, *any* messages, that are distinguishable as coming from one input verses another.

If you really must send something other than Continuous Controller messages, then you are an excellent candidate for Amelia's Compass upcoming **SceneChanger** software! Until that is available, many of the audio programs ControllerHub 8 talks to will have the ability to map one type of MIDI message to another. I highly recommend Qlab for this purpose, for example.

Here are descriptions of the user-editable parameters pertaining to each of the eight Pedal/Button Inputs:

MIDI Channel Number – all MIDI messages contain a MIDI Channel Number that can range from 1 to 16. Devices are assigned a MIDI Channel Number, and they can then ignore any messages they receive whose MIDI Channel parameter does not match their own.

MIDI CC Number – the MIDI protocol provides 128 Continuous Controller (CC) numbers. Many of these have suggested uses. For example, MIDI CC#7 is suggested for use as a means to send volume changes. But these are only suggested and therefore common usages; in fact we can use any CC number for any purpose we like, so long as the sender and the receiver both agree on the meaning. The purpose of the suggested uses is just to save a little work. For example, any device that has a user-editable volume parameter will almost certainly have mapped CC#7 to that parameter. Therefore we can generally assume that anything we send a CC#7 to will use that message to change its volume. Again, for our purposes we most often only need to make sure that each Pedal/Button input is assigned to a unique MIDI Channel / CC combination. We have 16 MIDI channels and 128 CC numbers available, so we have 2,048 unique combinations.

Max – this is used to limit or scale the range of numbers sent by the controller. For example, suppose you are controlling a parameter with a range of 0 to 100, instead of the usual 0 to 127. You would set max to 100, and now your pedal will output values from 0 to 100. This allows you to use the full range of pedal motion to sweep through a smaller range of numbers. You can even set max to 1, for switches that specify CC values of 0 or 1, instead of the usual 0 or 127. Check your target device's MIDI Implementation Chart to see what the range is for each parameter you wish to control.

Max is not normally needed when controlling parameters in software programs, because the scaling function is usually handled in the software.

Query – this is not really an editable parameter but a convenience feature[2]. Query sends a message to the ControllerHub 8 asking it to retransmit the current state of the controller connected to the input in question. Suppose you are controlling an audio software app that has presets. When you change the app's preset, it may not remember the last position of the controller in question. Some apps do and some app's don't. For those that don't, if you send a query message to ControllerHub 8 shortly after the preset change, ControllerHub 8 will retransmit the current controller position, synching the new preset with the physical hardware controller.

Calibrate – this is not really an editable parameter but a convenience feature[1]. Calibrate is used to make sure connected controllers are able to use the full range of the MIDI parameter they target.

When you send a MIDI Continuous Controller message, three bytes are sent. The first byte identifies the message as a CC message and also contains the target MIDI channel. The second byte identifies the CC number. The third is the value to which the CC target parameter is to be set.

Some expression pedals have some variability in their construction, such that some individual units can't cover the full range of CC values. You might connect an expression pedal and find that it can only cover the range from 0 to 114, for example, even though the full range is from 0 to 127. No matter how hard you press on the toe of the pedal, it won't go all the way to the top of the range!

The answer is to press the Calibrate button. This puts ControllerHub 8 into calibrate mode for that controller input. While in calibrate mode, move the pedal through its entire range, making sure to hit each endpoint at least once. Don't slam it into its endpoints; this will make it hard to reach them again later. Instead, just lightly touch each endpoint. While in calibrate mode, ControllerHub 8 will not send out any messages for the controller in question. It is internally keeping track of the World's Record High and World's Record Low that it sees as the controller moves. Now press the Calibrate button again, releasing ControllerHub 8 from calibrate mode for that input. When you move the pedal now, it will cover the full range of data values from 0 to 127.

Invert – this switch merely inverts the direction of pedals and buttons. A button that is normally off will be changed to one that is normally on by this switch (or vice versa). A movable foot pedal that sends increasing numbers toward the toe position will be changed to one that sends decreasing numbers toward the toe position.

Repeat – you might call this a "poor man's query". It's a brute-force way of keeping software synched with the latest physical pedal positions and button states. Every controller input that has the Repeat button engaged will retransmit its current position at regularly repeating intervals. The global repeat rate for controllers in repeat mode is set by the Repeat Rate control near the top left corner of the editor screen.

[2] All "real" parameters are saved to ControllerHub 8's nonvolatile flash memory. The states of the Query and Calibrate buttons are not saved to flash. So even though each controller input's configuration strip has two faders and seven buttons, there are only seven real editable parameters per controller input.

Button Mode – this switch activates Button Mode for the connected controller. It should generally be activated when an on/off footswitch is connected to the input, and deactivated when a variable expression pedal is connected (though there are exceptions to both cases). It should also be engaged when piano sustain pedals are connected. Any controller that has only two states should be used with Button Mode engaged.

The Button Mode switch activates an internal hysteresis function that keeps the physical button from chattering during transitions. Chatter isn't possible when the button is not being stomped on, but during the actual stomping action, a physical button can bounce multiple times. This happens in an almost imperceptible amount of time – just a few milliseconds – but results in additional false button presses unless the Button Mode switch is activated. For example, suppose you mapped the physical button to control a Preset Increment function in your software app. Without the Button Mode switch engaged, you might find that stepping on the button causes the preset to jump every so often by two or three instead of the intended single increment. I'm sure this could cause a lot of trouble during a live performance! The Button Mode switch is your friend. And you know it.

Regarding the difference between variable pedals and on/off switches, ControllerHub 8 doesn't really care whether a physical pedal or button is connected to its inputs. As far as ControllerHub 8 is concerned, a physical button is just a pedal with no middle section! So you can use a physical pedal in Button Mode – the button action will occur when the pedal crosses its halfway point. Conversely, you can also use a physical button for a pedal, but only the destination parameter's two extreme endpoints will ever be used. For example, you could hook a button up to a volume parameter and set the Toggle switch. The button would then act as a mute/unmute switch for that audio signal.

Or try routing a button to the sustain parameter of a synth or organ plugin, and activate the Toggle Mode switch. Now you can play a chord, hit the sustain button and use your hands and feet elsewhere while the chord is essentially being held down by ControllerHub 8. You can use this to solo over droning chords, for example. Later on, you can hit the button again to disengage the sustain function.

Toggle – this switch converts a physical momentary button into a toggle button. Physical buttons are of two types, momentary and toggle. Using the Toggle switch, we can convert a physical momentary button into a toggle-type button. The Toggle switch only has an effect when the Button Mode switch is also activated for the particular controller input.

But we can't go the other way; we can't convert a physical toggle button into a momentary button. There is no Un-Toggle Mode available! For this reason we recommend that you use physical momentary buttons, so that both options are available. This is a good time to recommend our ButtonPusher product!

Now if all you have on hand is a physical toggle button, it will work fine. But in this case do NOT engage the Toggle Mode switch. You'll end up with a double dose of toggling, and you'll have to press the button twice to get it to change states!

MIDI PatchBay Parameters:

Refer to the diagram showing the ControllerHub's internal connections. The factory default for all five of these switches is ON.

Pedals to MIDI – this switch allows Pedal/Button input messages to be sent out via MIDI Out.

MIDI to MIDI – this switch allows messages coming in via MIDI In to be relayed out via MIDI Out.

USB to MIDI – this switch allows messages coming in via USB to be relayed out via MIDI Out.

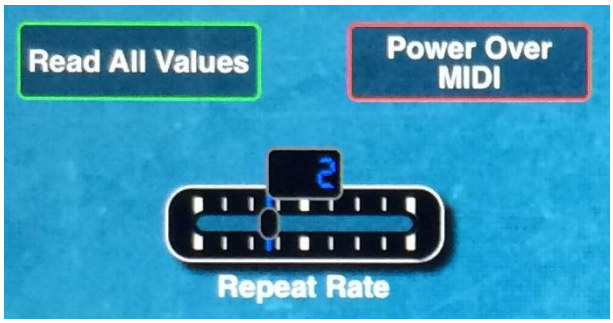
Pedals to USB – this switch allows Pedal/Button input messages to be sent out via USB.

MIDI to USB – this switch allows messages coming in via MIDI In to be relayed out via USB.

Known Bug: If you only use 5-pin MIDI out and not USB out, then you must disable Pedals->USB and MIDI->USB.

Miscellaneous Parameters:

Read All Values – Use this switch to read the values of all the ControllerHub's internal parameters and display them on the screen. This does not happen automatically upon USB connection, so it is highly recommended that you press it as soon as MIDI Designer makes contact with ControllerHub. Otherwise you will be seeing misleading information.



Power Over MIDI – Use this switch to send power out the MIDI In connector to the previous device in the MIDI daisy-chain. That device will obtain power through its MIDI Out jack. So MIDI messages flow from MIDI Out to MIDI In through the cable, but power flows backwards through this same cable. If this ControllerHub is being powered from a wall wart, or from the USB jack via a laptop or USB hub, this will work fine. But if this ControllerHub is being powered via USB from an iPad, it may not be possible to activate this function. The iPad generally can only source enough current to run a single ControllerHub.

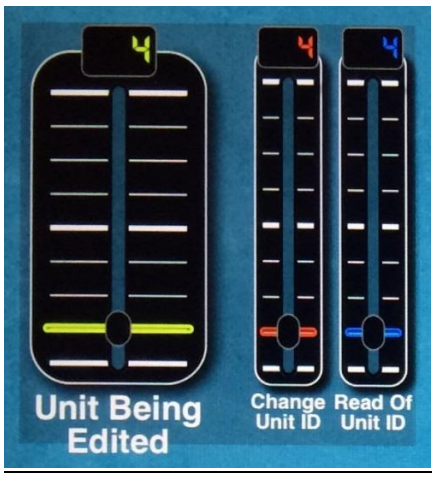
Repeat Rate – allows ControllerHub to repeat the state of any of its eight Pedal/Button inputs on a regularly recurring basis, even if they have not changed. This only applies to those controller inputs that have had their Repeat button activated in their configuration strip. The repeat rate you set here applies globally to all controllers in repeat mode (1 = 0.5Hz, 2 = 2Hz, 3 = 10Hz).

Reset to Factory Defaults – this will return all internal user-edited parameters to their "as new" states. But the reset button will only work if you first arm it:

Arm the Reset Button – this is a safety measure that makes it hard to accidentally press the reset button. If you really want to reset to factory defaults, first move this slider to its UP position. Then you'll be able to press the reset button. Unless the reset button is armed in this way, pressing the reset button will have no effect. It's probably a good idea to return the slider to its DOWN position after you've pressed the reset button.

Also remember that pressing the reset button returns the Unit ID to zero.

Parameters relating to Unit ID:



Change Unit ID – when you use more than one ControllerHub 8 and connect them through a USB hub (or if they are connected to multiple USB ports on your laptop), it becomes necessary for each one to have a unique Unit ID. The factory default sets this Unit ID to zero, which is the same as not having a Unit ID. But setting the ID to a non-zero number is easy to do. Make sure only one ControllerHub 8 is connected to the editor, and move the Change Unit ID fader to a new value. That's it! Connect each of your units individually to the editor, one at a time, and set each of their Unit ID's in this way. Make sure that they all have unique Unit ID's. After this has been done, they can be connected together and the editor can individually address them.

Unit Being Edited – when multiple ControllerHub 8's are connected via USB Hub and they've been set to unique Unit ID's, you need a way to individually address them for editing purposes. This control is how you do that. One important additional fact is that if you set this control to zero, it puts the editor in Broadcast Mode, meaning all connected units must act on editor commands, regardless of their Unit ID. This is very useful when you forget what ID you've assigned to a unit and you need it to return its ID to you. You put the editor in broadcast mode by setting the Unit Being Edited to zero, and then pressing Read All Values.

Read Of Unit ID – when you have multiple ControllerHub 8's connected via USB Hub, you sometimes need a way to verify what each unit's ID has been set to. When you press the Read All Values button, the "Read of Unit ID" fader will be set to the connected unit's ID. But there is a confusing caveat, so read carefully: You can only reliably do this when there is only one unit attached, and you set the Unit Being Edited to zero.

Automatic Save to Flash Memory

During an edit session, if any of ControllerHub 8's internal parameters are changed, these changes are automatically saved to internal flash memory after 10 seconds have elapsed without further changes. You can tell this has happened when the LEDs flash in sequence. This ensures that when you power the unit off, your parameter settings will still be there when you turn it back on. But don't power down before the 10 seconds have elapsed! If the LEDs haven't flashed, your edits haven't been saved yet. So after a parameter edit, wait for the LEDs to flash and THEN power down.

Editing parameters using MIDI SysEx messages

Here's how to talk to ControllerHub via MIDI in order to edit its internal parameters. We have a MIDI Designer template all ready to go for this purpose, but you are free to use whatever means you have at your disposal to send these MIDI messages. Alternately, you can create your own MIDI Designer template to do the job, or start with our MIDI Designer template and edit it to your liking.

All of our MIDI messages are the same length – 10 bytes. There's no IF/THEN/ELSE logic to determine the length of messages (unlike MIDI itself!). They are all System Exclusive MIDI types, and the only thing that changes from message to message is the target parameter number and of course the actual new parameter setting.

Here is a table showing the MIDI System Exclusive message format and the meaning of each byte (the dollar sign $ indicates that values are in hex):

First we have the Start-of-SysEx header byte:
$F0

Then we have the Amelia's Compass MIDI Manufacturer's ID bytes:
$00
$02
$01

We were late to the game by 30 years, so we don't have one of the cool single-byte ID's that were issued back when USB was still a twinkle in Silicon Valley's eye and MIDI was implemented with vacuum tubes and cloth-covered wire inside tweed enclosures, while kids did Duck-and-Cover drills under their desks. But hey, our 3-byte ID is nice and low; that's worth something, isn't it?

Next comes the ControllerHub Product ID:
$01

We're hoping to have more products with their own ID's soon. We don't want ControllerHub to be an only-child.

Then we have a Unit ID byte which can be used when you have more than one ControllerHub 8:
$0*x* (where *x* is a number from 1 to 32, or zero)

See the section More on Unit ID's to find out how to use these.

Next comes the Parameter Group ID:
$0*x* (where *x* is a number from 0 to 8, used to select one of nine internal targets for the message being sent)

$00	Pedal or Button Input 1 – we want to edit one of the parameters associated with TRS Input 1
$01	Pedal or Button Input 2 – we want to edit one of the parameters associated with TRS Input 2
$02	Pedal or Button Input 3 – we want to edit one of the parameters associated with TRS Input 3
$03	Pedal or Button Input 4 – we want to edit one of the parameters associated with TRS Input 4
$04	Pedal or Button Input 5 – we want to edit one of the parameters associated with TRS Input 5
$05	Pedal or Button Input 6 – we want to edit one of the parameters associated with TRS Input 6
$06	Pedal or Button Input 7 – we want to edit one of the parameters associated with TRS Input 7
$07	Pedal or Button Input 8 – we want to edit one of the parameters associated with TRS Input 8
$08	Internal MIDI Patch Bay / Misc. – we want to edit one of the parameters associated with MIDI routing, or change one of the miscellaneous functions.

Finally, we send the Parameter ID and the Parameter Value, each in its own byte. The exact meaning of the Parameter Value byte changes depending on the Parameter Group and Parameter ID selection, as follows:

For Pedal/Button Inputs:

Parameter ID:		Parameter Value:
$00	MIDI Channel assignment	$0*x* (where *x* is a number from $0 to $F)
$01	MIDI Continuous Controller assignment	$*xx* (where *xx* is a number from $00 to $7F)
$06	Invert switch	$00 or $7F (for OFF or ON)
$07	Button Mode switch	$00 or $7F (for OFF or ON)
$08	Toggle Mode switch	$00 or $7F (for OFF or ON)
$09	Auto Repeat switch	$00 or $7F (for OFF or ON)
$0B	Calibrate Mode switch	$00 or $7F (for OFF or ON)
$0C	Query button	$00 or $7F (for OFF or ON)

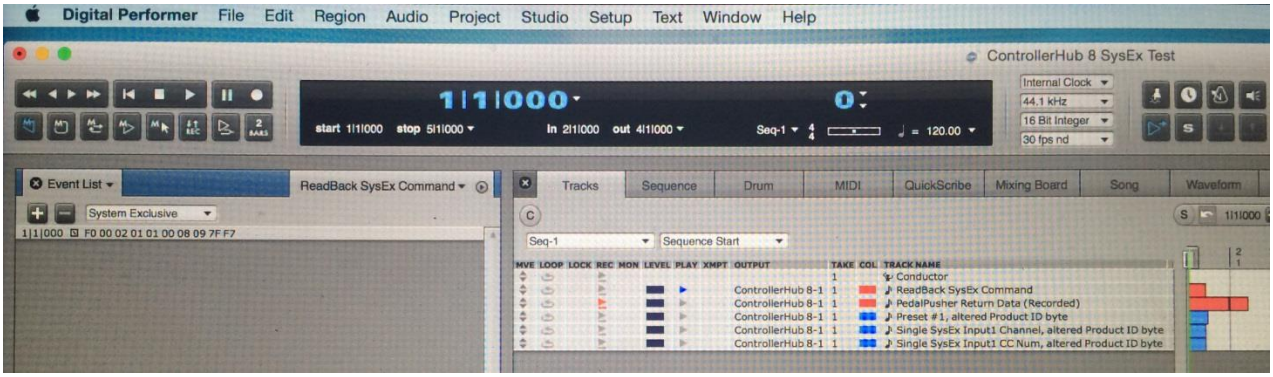
For the MIDI Patch Bay (and other miscellaneous parameters):

Parameter ID:		Parameter Value:
$00	Pedal/Button Inputs to MIDI switch	$00 or $7F (for OFF or ON)
$01	Pedal/Button Inputs to USB switch	$00 or $7F (for OFF or ON)
$02	MIDI Input to MIDI Output switch	$00 or $7F (for OFF or ON)
$03	MIDI Input to USB Output switch	$00 or $7F (for OFF or ON)
$04	USB Input to MIDI Output switch	$00 or $7F (for OFF or ON)
$05	Power-Over-MIDI switch	$00 or $7F (for OFF or ON)
$06	Global Auto-Repeat Rate	00 = 0.5 Hz, 01 = 2 Hz, 02 = 10 Hz
$07	Unit ID	$00 to $1F
$08	Verilog Version Number LS	$00 to $7F
$09	Verilog Version Number MS	$00 to $7F
$0A	Read All Values	Any (not used)
$0B	Reset Defaults	$00 or $7F (for OFF or ON)

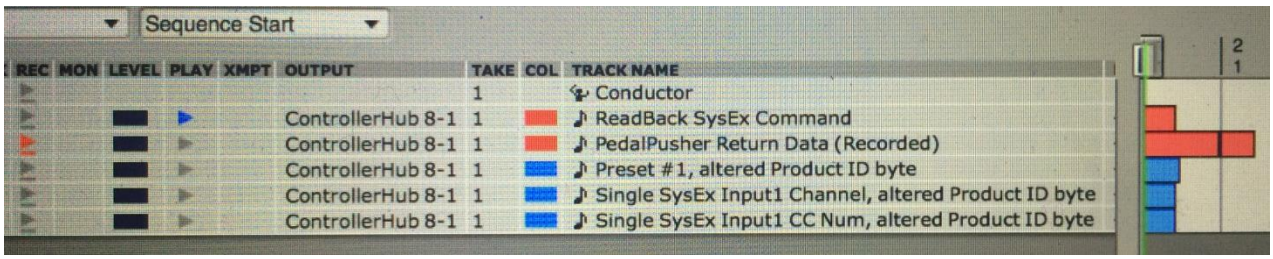
The whole message is terminated with an End-of-SysEx byte:
$F7

Using a MIDI Sequencer to edit the ControllerHub internal parameters:

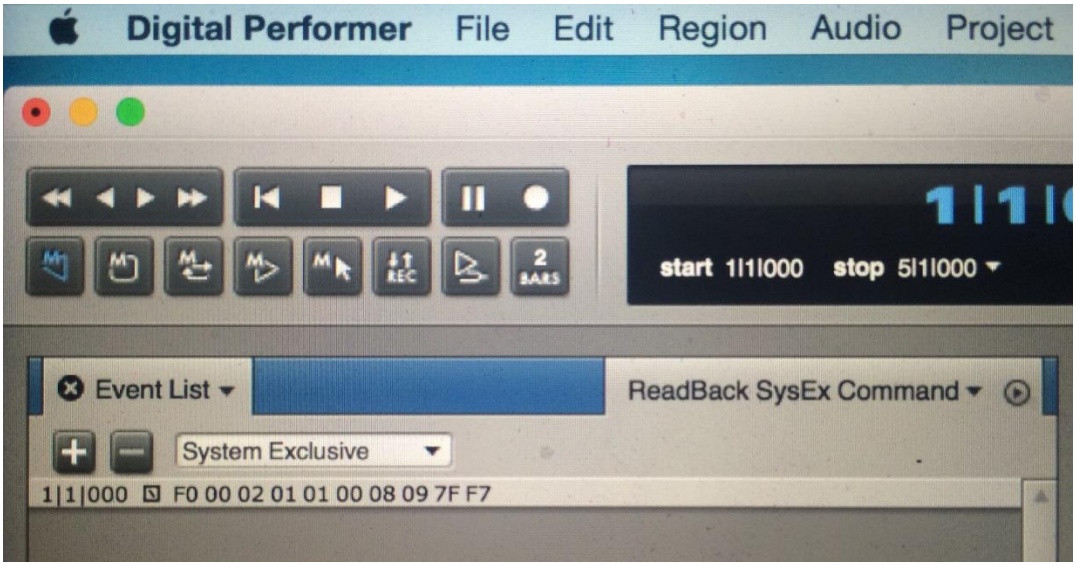
To show you how you can use anything capable of generating MIDI SysEx commands to edit ControllerHub 8's internal parameters, we've created a project to do so in Digital Performer.



The two tracks in red are the ones we're interested in. Let's zoom in on those:



The first is a MIDI track called "Read All Values SysEx Command". If we open that track in the Event List, we see that it contains one MIDI SysEx message:



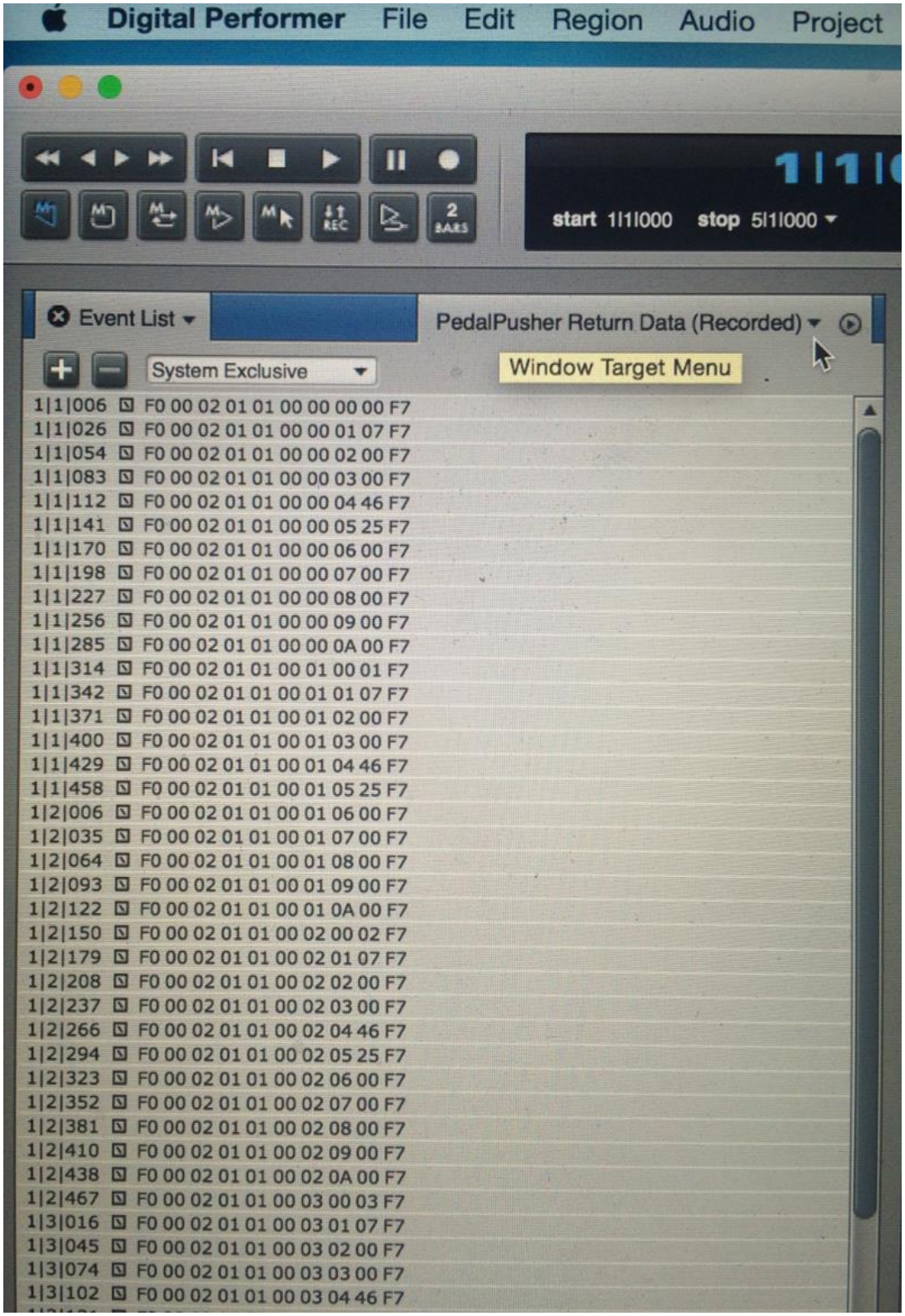
Let's parse that SysEx message:

F0	Start of SysEx
00, 02, 01	Amelia's Compass Company ID
01	ControllerHub 8 Product ID
00	Unit ID (set to zero for Broadcast Mode)
08	Parameter Group = 8, MIDI Patch Bay & Misc. Parameters
0A	Parameter Number = A, Read All Values button *(the screen shot is old, it shows 9)*
7F	Parameter Value = 7F, meaning ON
F7	End of SysEx

So here we have the correct "Read All Values" command to send to the ControllerHub 8, which will cause it to send back all of its parameter settings in one blast.

You'll notice that the "Read All Values SysEx Command" track is play-enabled, so it will send its MIDI to the connected ControllerHub8. Meanwhile the "ControllerHub 8 Return Data" track is record-enabled, so we can capture the return data and get a look at it.

All we have to do is hit the sequencer's Record button. We see the ControllerHub 8's Rx LED blink briefly followed by a long blast on the Tx LED, corresponding to the long blast of return data. Activity stops after a couple seconds, so we can hit the sequencer's Stop button and look at all the captured return data:



Notice that the return data is also made up of a long list of individual MIDI SysEx commands, so you can send this data blast (or any of its individual commands) back to the ControllerHub 8 at any time.
You can copy and save this return data to a different MIDI track for editing. Any of its SysEx messages can be edited, and the entire edited batch sent back to ControllerHub to change its internal parameters.

In each SysEx message, the fourth-to-last byte gives the Parameter Group, the third-to-last byte gives the Parameter Number, and the second-to-last byte gives the Parameter Value. Generally, it is the Parameter Value you will change to achieve your desired functionality.

In fact you can duplicate the above track to any number of other named tracks, and each of these could be edited to contain a different downloadable ControllerHub 8 preset.

To send one of these presets back to the ControlHub 8, make sure that it is the only play-enabled track and hit Play on the sequencer. If more than one track is play-enabled, things will likely not work.

It is not possible to "brick" the ControllerHub 8. You can always reset it to the factory defaults.

Advanced: Notice the fourth-to-last column of bytes, which indicates the Parameter Group. Reading down this column, you can see that all the Group 0 parameters are returned first, followed by all the Group 1 parameters, then the Group 2 parameters, etc. If you were to scroll down, you'd finally see the Group 8 parameters at the end of the list, making nine groups in all. The last group contains the global parameters, whereas the first eight groups are for the controller inputs.

If you look closely at any of the input groups, you can see that more parameters are returned than are listed in the MIDI Implementation Chart or described in this manual so far. Specifically, these mystery parameters are numbered 02, 03, 04, and 05. These are the calibration factors for each of the inputs. If a pedal doesn't hit the endpoints and you recalibrate it to do so, you'll see these numbers change. They're returned in the Read All Values blast because they're part of the data that gets saved to flash.

It won't hurt to resend these back to the ControllerHub 8, because the calibration settings are ignored if you try to force them back into the ControllerHub 8 via SysEx command. The only way to change them is to go through the approved calibration method. Therefore, you might wish to delete these commands if you save the long SysEx blast to a named preset MIDI track. But again, it's also fine if you leave them in.

MIDI Implementation Chart (with SysEx hex values used to send each message)

Saved to Flash?	SysEx Start	Amelia's Compass ID		Product ID	Unit ID	SysEx Channel	Param Group	Param Num	Default Param Val	SysEx End	Description	Range	Note
X	F0	00	02	01	01	x	00	00	00	F7	Input 1, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 1, Continuous Controller assignm	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 1, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 1, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 1, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 1, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 1, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 1, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 1, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	01	00	01	F7	Input 2, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 2, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 2, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 2, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 2, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 2, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 2, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 2, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 2, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	02	00	02	F7	Input 3, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 3, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 3, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 3, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 3, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 3, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 3, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 3, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 3, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	03	00	03	F7	Input 4, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 4, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 4, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 4, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 4, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 4, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 4, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 4, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 4, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	04	00	04	F7	Input 5, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 5, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 5, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 5, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 5, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 5, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 5, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 5, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 5, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	05	00	05	F7	Input 6, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 6, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 6, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 6, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 6, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 6, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 6, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 6, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 6, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	06	00	06	F7	Input 7, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 7, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 7, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 7, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 7, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 7, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 7, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 7, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 7, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	07	00	07	F7	Input 8, MIDI Channel assignment	00 to 0F	
X	F0	00	02	01	01	x		01	07	F7	Input 8, Continuous Controller assignr	00 to 7F	
X	F0	00	02	01	01	x		06	00	F7	Input 8, Polarity	00 = Normal, 7F = Inverted	
X	F0	00	02	01	01	x		07	00	F7	Input 8, Button Mode switch	00 = Expression Pedal, 7F = Button	
X	F0	00	02	01	01	x		08	00	F7	Input 8, Toggle Mode switch	00 = Momentary 7F = Toggle	(No effect if Button Mode not active)
X	F0	00	02	01	01	x		09	00	F7	Input 8, Auto-Repeat switch	00 = Normal, 7F = Repeat	
X	F0	00	02	01	01	x		0A	7F	F7	Input 8, Maximum Value	00 to 7F	
	F0	00	02	01	01	x		0B	00	F7	Input 8, Calibrate Mode switch	00 = Normal, 7F = Calibrate	
	F0	00	02	01	01	x		0C	00	F7	Input 8, Query button	00 = Idle, 7F = Query	
X	F0	00	02	01	01	x	08	00	7F	F7	Pedal/Button Inputs to MIDI switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		01	7F	F7	Pedal/Button Inputs to USB switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		02	7F	F7	MIDI Input to MIDI Output switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		03	7F	F7	MIDI Input to USB Output switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		04	7F	F7	USB Input to MIDI Output switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		05	00	F7	Power-Over-MIDI Switch	00 = OFF, 7F = ON	
X	F0	00	02	01	01	x		06	02	F7	Global Auto-Repeat Rate	00 = w, 01 = x, 02 = y, 03 = z (msec)	
X	F0	00	02	01	01	x		07	00	F7	Unit ID	00 to 1F	
X	F0	00	02	01	01	x		08		F7	Verilog Version Number LS		(Read only)
X	F0	00	02	01	01	x		09		F7	Verilog Version Number MS		(Read only)
	F0	00	02	01	01	x		0A	00	F7	Read All Values	00 = OFF, 7F = ON	
	F0	00	02	01	01	x		0B	00	F7	Reset Defaults	00 = OFF, 7F = ON	

Specifications

Dimensions:	4.5” wide x 1.75” tall x 4.325” deep
Weight:	0.428 kg (15.1 oz.)
Wall Wart:	Use a wall wart that supplies a DC voltage within the range of 9 to 24V, with the positive voltage on the inside center conductor.
Power Consumption[3]:	750mW @ 24V input (31.25 mA), 550mW @ 10V input (55mA), 400mW @ 5V input (80mA)

[3] The power consumption of the ControllerHub 8 depends on a couple things:

The higher the input voltage from the wall wart, the less efficient the internal power conversion. But we specify a range of wall wart voltages to increase the likelihood you’ll already have a suitable one on hand. Each pedal or button you plug in to the ControllerHub 8 also causes a bit more power to be consumed.

For purposes of specifying a worst-case number, we assume the maximum recommended 24V wall wart voltage and all eight inputs connected to expression pedals. Under these conditions, the ControllerHub 8 still consumes less than 750 milliWatts (mW). It’s not a lot of power – for example you’d have to hook up 67 ControllerHub 8’s to use as much power as a 50W light bulb.

Using a 10V wall wart, the power consumption drops to 550mW (55mA). When using PMIDI or USB to phantom-power the unit, its power consumption drops to 400mW (80mA).

Revision History

(Not tracked until 1v8)

1v7 -> 1v8

Corrected info left over from ControllerHub 4 on p.24, talking about Parameter Group ID on p. 24. It still said this number ranges from 0 to 4. Corrected to say 0 to 8.

1v8 -> 1v9, 5_17_21

Added length of SysEx messages – 10 bytes (p.23)
Bug disclosure (p. 20)
Corrected/added rev level in page header