Welcome to Organelle™

The User Guide for a New, Multipurpose Digital Instrument from Critter & Guitari®

Critter & Guitari

This user guide and its content is subject to change without notice. It should not be viewed as a commitment by Critter & Guitari Inc.

Though we aimed for accuracy and clarity herein, Critter & Guitari Inc. is not liable or responsible for errors or inaccuracies occurring in this user guide.

This user guide was written by Dave Linnenbank.

Second Edition, May 2016

All specifications are subject to change without notice: users should check that they have the most current manual.

For warranty information, please view the document that shipped with your Organelle.

Table of Contents

Chapter Zero

Quick-start Guide

Let's get sound. Now!

Welcome to the world of Organelle! You seem to be in a hurry so here is the short version for getting up and running…

Package Manifest

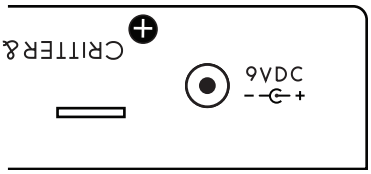
Along with this card, there are two main items in the box you just opened.

- **Your Organelle.** Also note that a tiny white USB drive is also plugged in on the right side of the unit. This drive houses the patches that will bring Organelle to musical life.
- **Its power adapter.** While the power supply has prongs for US-style wall sockets, it will work with input voltages from 100 to 240VAC at 50/60Hz. You may need an adapter for the shape of your wall socket.

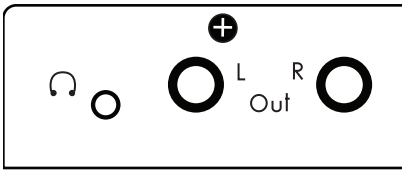
Getting Started

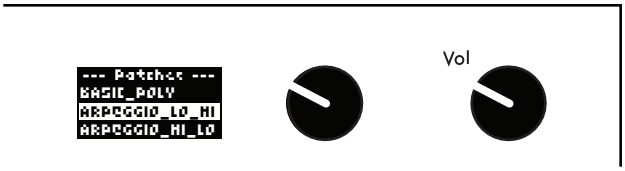
For the simplest configuration, follow these steps.

1. **First connect the power adapter to a power outlet, and then connect its plug to the leftmost port on the back of the Organelle.** Once the Organelle is receiving power, it will begin booting up. The screen will indicate when this process is done.



2. **With the volume all the way down, connect your headphones or audio output.** The knob furthest on the right of Organelle controls its output **`Vol`**(ume). Start with this knob all the way to the left (no sound). From the front of the Organelle, peer over the back of the unit (don't worry; the port labels will be upright from this vantage). The leftmost 1/8" jack is for headphones, and the following two 1/4" jacks are for left and right audio output, respectively.





3. **Load a patch.** Immediately to the left of the volume knob is the *Selector* encoder. Turn this encoder to select one of the listed patches, and then press down the top of the encoder to load the patch.

4. **Play!** The wooden keys that are arranged in a piano-layout represent two octaves of notes. Play them, adjust some parameters (via the four knobs on the left), and enjoy! And if you want to try a different patch, simply turn the Selector encoder and select another patch.

5. **When you have had enough fun for now, properly shut down Organelle.** This is achieved by turning the Selector encoder so that the menu screen appears again. From here, scroll upward (by turning the encoder to the left) until you go past the **`PATCHES`** and reach the **`SYSTEM`** options. Select the first option, **`Shutdown`**, and then press down the encoder to engage this function. Finally, Organelle's display will notify you when it is safe to unplug the power supply from the unit.

Chapter One

Organelle™ Concepts

What is this pretty thing?

Again, welcome to the world of Organelle! As this instrument can work for people in many different ways and at multiple depths of operation, let us begin by getting a few ideas straight, starting with the most obvious question…

What is Organelle?

We can (and will) talk about what the literal Organelle device *is*, but we'd do better to start with what it *can be*.

Organelle is an instrument.

As you may have already noticed in the quick-start guide (chapter zero), Organelle can be connected quickly and is ready for sound. Other than the Organelle itself, its included USB drive and power supply, and a sound-producing device (such as headphones or an audio cable to connect to a sound system), no additional peripherals are needed for performance.

Organelle is an effects processor.

In addition to audio output ports, Organelle also has an audio input port. Accordingly, patches can access and make use of incoming audio in various ways. This can range from a simple effect processor (like a filter or basic delay) to something more elaborate (like a sampler or vocoder or something else entirely).

Organelle is a generator.

There is no requirement that the patches you load into Organelle are triggered by playing notes on the device. You could create patches that simply drone and/or create evolving textures on their own over time. The audio input could even be used as a source. As this definition is starting to become circular, let's cut to the chase…

Organelle is whatever you want it to be.

In truth, Organelle is a vessel for your musical ideas, connecting your own desires for musical expression with customizable technology and portability. You may use Organelle in a completely different way than someone else, and that is not just okay: it's the entire point.

So is it hardware or software?

In short, yes. Organelle is a hardware device that comprises both controller elements (the ports, knobs, keys, etc.) and a modern microcomputer housed inside the case. The microcomputer itself is running a version of the Linux operating system, and one use of the USB ports is to connect a drive that contains your software patches. (The USB drive included with your Organelle is preloaded with the factory patches and ready to go.)

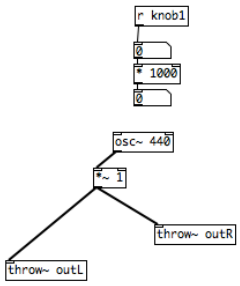
What are these "patches" you speak of?

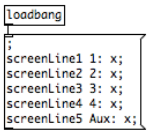
They are files configured with the program *Pure Data*. While the term "patch" often refers to the settings and parameter values that create one sound in a synthesizer (or some other predefined system), Pure Data patches are a bit more expansive. Each patch represents the entire software system for taking any/all input received by Organelle, processing it as desired, and then delivering the output as audio, etc., via Organelle's output ports. (So by analogy, these patches are closer to both the synthesizer structure itself *and* all the settings and parameters that define its initial sound.)

Some patches require various support files (audio media, other support patches that are being referenced, "external" objects, etc.). Any time we discuss a particular patch, it is fair to assume that we are also referencing any necessary subsidiary files.

What is Pure Data? And do I need to learn it to use Organelle?

Pure Data (often called *Pd* for short) is a visual multimedia programming environment, meaning that its software files (yes, those *patches*) are created by adding objects from its library and then interconnecting them with virtual *patch cords*. For example, one simple Pure Data patch looks like this.





And no, you do not need to learn Pure Data to use Organelle. Aside from the factory patches, you can also find a repository of additional patches at organelle.io for your perusal. You *can* customize or create your own patches — Pure Data is free and available for all computer platforms (see chapter five) — but "to code or not to code" is completely up to you.

What other concepts may be useful to understand?

A basic understanding of audio can only help. And MIDI (musical instrument digital interface) is the protocol for triggering *notes* and sending *control messages*. To use Organelle as is, basics are enough. If you decide to create patches, a little bit more will be required, but we will get to all that in later chapters.

How to Use This Manual

Certain chapters (such as this one!) are relevant to everyone. But depending on how you will start using Organelle, some chapters may be more valuable to you than others.

- Regardless of your intentions, the quick-start (chapter zero) and concepts information (chapter one) will benefit you.
- If you are satisfied with the included patches alone, the information on general hardware configuration (chapter two), system operation (chapter three), and the factory patch listing (appendix A) will all be relevant to you.
- If you are looking to load additional patches into Organelle, then chapter four will also be useful to you. (And again, visiting organelle.io would be a good place to start your patch search.)

- If you want to edit patches or even create some patches of your own, chapter five will be essential. And if you plan on working off of the internal microcomputer itself, chapter six will need some review.
- Finally, the various appendices can be useful to everyone.

And do realize that your uses of Organelle are likely to change over time. If a chapter is not important to you today, don't feel bad about that: the chapters are happy to wait for you.

Chapter Two

The Hardware Unit

Acquainting yourself with the box.

As we begin to explore the universe that Organelle makes available to us, we should start with Organelle's place in the physical universe: its hardware.

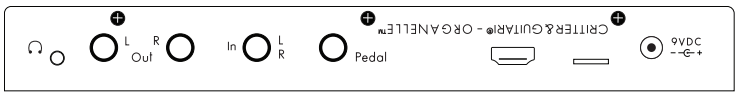
In this age of software and "virtual" devices, please do not neglect the hardware side of things! While its microcomputer does indeed run everything that Organelle does, we endeavored to make Organelle an "un-computer." Configuring and then interacting with the physical interface is just as critical as anything you do on the software side. In fact, this is how the pretty box can become an instrument…

Layout of the Unit

Organelle is rectangular, with three of its six faces containing either controls, ports, or other interface items. We will start with the back and right-side panels (where all the ports are housed) before moving to the controls of the main face.

Back Panel

If you have turned on Organelle, then you already have some familiarity with its ports, but there is a little more here than you realize (and a little more than is labeled).



Again, our orientation would be upside down if you walked around to the back of the unit and directly faced the back panel. We are assuming that you standing in front of Organelle, just as you will be when operating it. From that position, you would access the back panel either by leaning your head forward or by tilting Organelle upward.

- The headphone port (labeled with the icon, 🎧) is an 1/8" TRS (stereo) jack. It delivers the stereo audio output of your current patch, as scaled by the **`Vol`**(ume) knob.

- The **L**(eft) and **R**(ight) **Out**(put) ports are both 1/4" TS (mono) jacks. They deliver the left and right audio outputs of your current patch, as scaled by the **Vol**(ume) knob.
- The single **In**(put) L/R port is a 1/4" TRS (stereo) jack. It receives any stereo audio input that you would like fed into your current patch.

NOTE: If a 1/4" TS cable is connected, any incoming signal will only be received by the left input.

- The foot **Pedal** port is a 1/4" jack. It is intended to be connected to a keyboard sustain-/damper-style pedal, which will deliver on/off messages to your patch.

NOTE: Organelle presumes that any sustain/damper pedal used has a "normally closed position" (negative polarity).

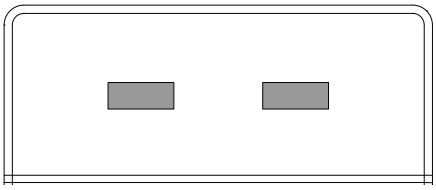
NOTE: If an expression pedal is connected, your patch should receive a continuous range of values. In our experience, various expression pedal models and settings tend to deliver different ranges of values. This can be addressed at the patch level.

- The HDMI® port delivers the video output of Organelle's internal microcomputer. [For additional information on using the HDMI port, see chapter six.]
- The microSD card slot contains a card that acts as the internal microcomputer's root disk. We do not recommend ejecting or otherwise manipulating this card as nothing good will come of it.
- The power port (labeled **9VDC**) is for connection to Organelle's own power supply.

NOTE: The output specifications of this power supply are: 9VDC, 1000mA, and a tip with center-positive polarity (). Any power supply used with Organelle must meet these three specifications.

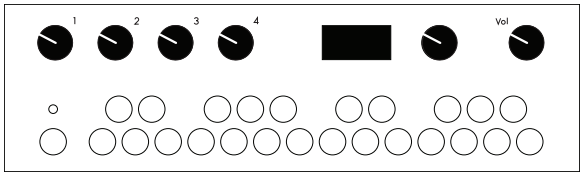
Right-side Panel

Compared to the back panel, the right-side panel is downright simple, housing two USB 2.0, Type A



ports. One of these identical ports must be used for connecting a USB drive that houses your Pure Data patches (such as the one included with Organelle). They can also connect class compliant devices that utilize MIDI over USB or other computer peripherals.

Main Face



The main face is both Organelle's primary interface with you and the place that you will spend the most time. While the other panels are critical when Organelle is being setup or when you are altering its configuration, the main face is the operations center for when you are actively running the show.

- Knobs **1**, **2**, **3**, and **4** are available for parameter control within your patch. Each knob is typically assign to a parameter that is then continuously altered across a preset range of values by movement of that knob. Movement of each knob can also send a corresponding continuous controller (CC) MIDI message. [For information on the default MIDI operation of Organelle, see chapter three.]
- Organelle's display provides a window into its microcomputer brain, serving as the on-board method of monitoring and adjusting both the system itself and your patches.
- The *Selector* encoder accompanies Organelle's display because they are dependent upon one another. While a patch is loaded, turning the Selector causes the display to show the menu screen. By leaving the Selector alone for a few seconds, the display will revert to the patch information screen.
- The **Vol**(ume) knob governs the potential audio output level of Organelle. The knob ranges from silence (in audio terms, -∞) at the far left to no attenuation (unity gain) at the far right. Any adjustments to the Volume knob take effect immediately.
- The maple key at the far left and its accompanying LED comprise one special unit: the *Aux button*. By default, the Aux button does nothing, but each patch can be configured to use the input from the key for any type of mode switch or anything else. The LED has eight static states (off plus seven color options) and is generally used to provide the user with visual feedback of the Aux button's status. As with so much about patches, the function of this control will be anything the patch designer deems appropriate.

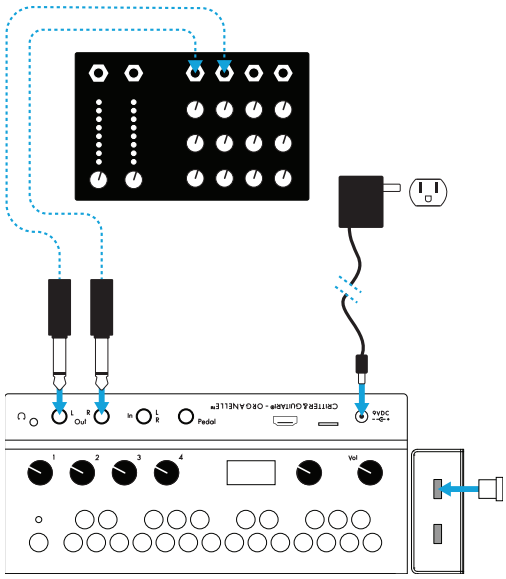
- After the Aux button, the 24 other maple keys work together as a group. As their piano-style layout may have indicated, these keys are for playing notes. By default, each key triggers a "note on" MIDI message when it is pressed down and a corresponding "note off" MIDI message when it is released. For patches that use note messages to trigger or affect audio output, these keys will be your primary performance vehicle. [For information on the default MIDI operation of Organelle, see chapter three.]

A Few Configuration Ideas

While Organelle is an open platform that permits and encourages nontraditional setups, we will now look at a few potential configurations for Organelle. Rather than suggested setups, treat these more as baselines or ideas; nearly all elements of any configuration can be mixed and matched.

Minimal Performance Setup

Here we start with a variation on the setup proposed in the quick-start guide (chapter zero). This bare-bones approach is the most compact performance configuration possible.

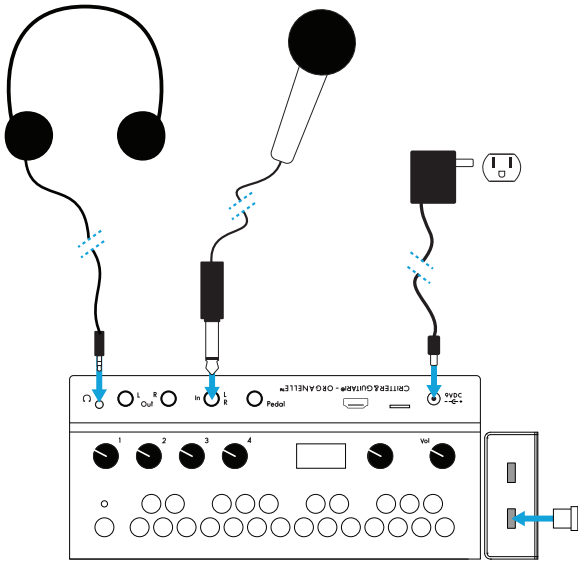


Note that the power is connected to the wall and that the first USB port has a flash drive inserted with our **`Patches`** folder. Without both the power adapter and USB drive attached, Organelle cannot operate and run patches, meaning that you cannot do anything of use. Accordingly, every possible configuration will contain these two items.

The **L**(eft) and **R**(ight) audio **`Out`**(put) ports are connected as a stereo pair to a mixer, which assumably runs to the venue's PA system, etc. (Instead of going straight to a mixer, these ports could just as appropriately be connected to direct boxes [DIs].) If there is a sound person controlling levels, you may want to leave the Volume knob all the way up, providing maximum signal for them to work with.

Audio Input from a Microphone

This example uses a microphone as an audio source for Organelle and headphones for audio output (to avoid feedback). This setup is also fairly minimal.



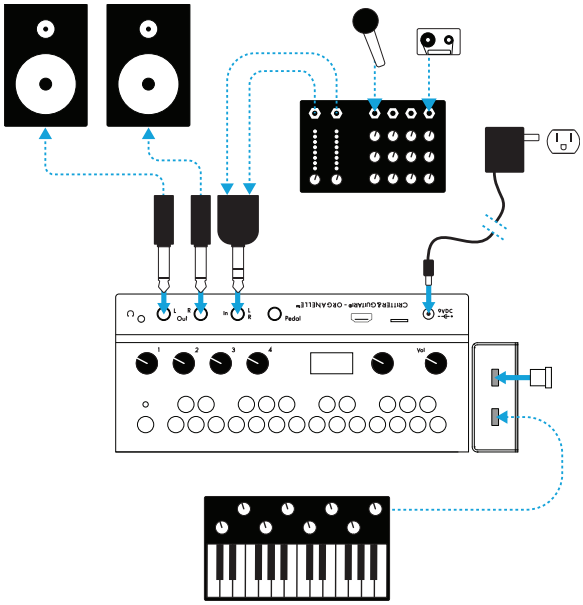
As the audio **In**(put) L/R port accepts a 1/4" cable, you will need something other than a regular XLR cable to use a microphone with Organelle. This could mean a cable with the appropriate connection for your microphone (probably XLR) and a 1/4" plug on the other end for Organelle, a standard XLR cable with a female XLR to 1/4" adapter attached, etc.

As most microphones are monophonic, most patches that use audio input are likely to either sum the left and right inputs together or only use the left input. This is not problematic in and of itself, but it could affect your setup choices and expectations.

And while we will no longer note the inevitable presence both of the power adapter and the USB drive (containing our **Patches** folder), we will observe here that our USB drive is now connected to the second USB port instead of the first one. The point is that Organelle's two USB ports are completely interchangeable.

A USB MIDI Controller and Audio Input from a Mixer

This example uses powered speakers for audio output, a mixer as the source of audio input, and a USB MIDI controller to supplement Organelle's own keys and knobs.



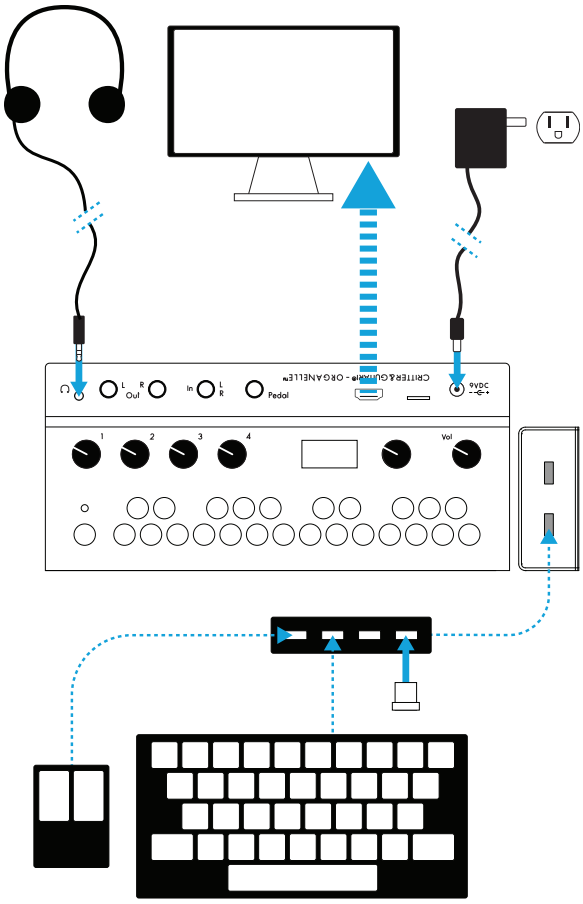
By using a mixer for Organelle's audio input, we can now use multiple audio sources with Organelle, and we can also use the mixer's various gain controls as input level adjustments for Organelle. As most mixers provide two mono outputs and Organelle has a single stereo input,

you will likely need an adapter or insert cable (aka, a Y-cable) for connecting the two devices.

In the case of using powered speakers, you need only connect each of Organelle's audio outputs to one of your speaker's inputs.

In addition to our trusted USB drive, a MIDI controller is connected to the other USB port. Any incoming MIDI messages are sent directly to the current patch, which will determine how to use them. As long as it is class compliant, any device supporting MIDI over USB would work, whether it is an 88-key piano-style controller or a DJ-style controller, etc. Even MIDI-only controllers will work when connected to Organelle via a USB class compliant MIDI interface.

An External Monitor, a USB Hub, and Computer Peripherals



This example uses headphones for audio output, a USB hub for connecting more than two devices, and an HDMI monitor for seeing Organelle's microcomputer at work.

Connecting an HDMI monitor to Organelle shows the internal microcomputer's command-line interface and graphical user interface. [For additional information on using the HDMI port, see chapter six.]

Our USB drive is still connected, but it is now running through a class compliant USB hub. By adding a hub to our configuration, the number of USB devices we can connect grows to the capacity of the hub. Connecting an HDMI monitor assumes that you want to operate Organelle's microcomputer in a typical way, and this implies the use of peripherals, such as a mouse and keyboard. This necessitates the use of a USB hub. And even without using an HDMI monitor, a USB hub would be needed if you wanted to connect multiple MIDI controllers at once, etc.

Chapter Three

Operating Organelle™ by Itself

Using Organelle's internal operating system.

As we stated early on and as the configuration ideas have shown, Organelle is a fully capable stand-alone instrument. To use Organelle in this fashion requires understanding the workings of its internal operating system. By learning the options provided by Organelle's software along with the uses of the Selector encoder and the on-board display, you will be ready to travel and perform with Organelle alone.

In this chapter, we'll explore where this combination of the Selector and on-board display can go, and we will also go through the default MIDI operation of Organelle and how connected USB MIDI devices interface with Organelle.

These topic areas will prepare you for general use of Organelle. So let's start using it now and get the pretty box singing.

The Selector and the On-board Display

To interface with Organelle, we will primarily work with two of its components. The Selector encoder allows us to navigate system options and execute functions, and the on-board display shows us the choices we have and provides feedback on our current patch and system.

In order of appearance, we will cover the two primary screens that peek out at us from the on-board display.

Menu Screen

When Organelle is powered up, it first runs through its boot sequence and then drops us into its *menu screen*.

The menu screen itself comprises two sections: the system section (or system menu) at top and the patches section (or patches menu), which we see in the above image.

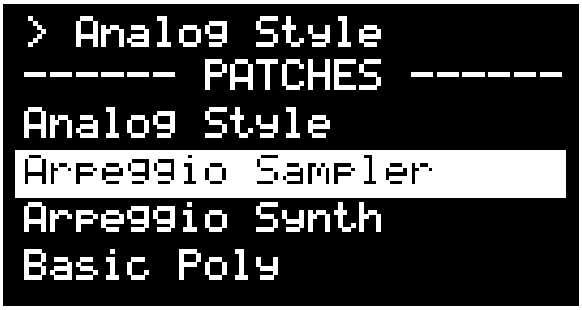
Patches Menu

After Organelle first boots up, it places us directly in the *patches section*, which is helpfully labeled **PATCHES**. By turning the Selector to the left and right, we are able to move up and down respectively in the patch list.

(If we scroll too far to the left, we will navigate past the patches menu and up into the system menu. In that case, simply move back down to the patches menu.)

To load the selected patch: press down on the top of the Selector encoder.

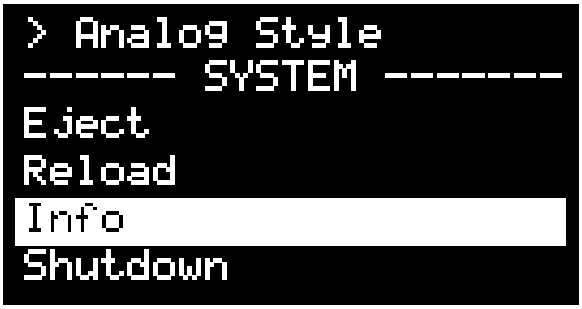
After the patch is successfully loaded, Organelle's display shifts to show us the patch information screen, which we will discuss in a moment. To return to the menu screen, turn the Selector. The only difference is that the active patch is now displayed in the top line of the menu screen.



NOTE: One neat performance trick is that Organelle remembers its selection in the patches menu. So if you want to cue up the next patch you will need, you can turn the Selector and leave that patch selected. Now all you have to do to load that patch is press down the Selector knob — you don't even have to turn the Selector or call up the patches menu.

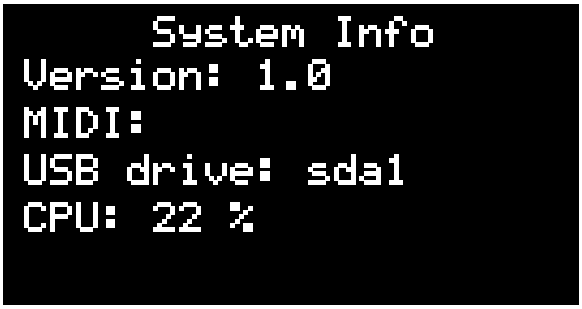
System Menu

At the top of the menu screen is the *system menu*. The top of this section is labeled **SYSTEM**, and it contains four options.



To engage one of these options, follow the same procedure as loading a patch: select the desired option by turning the Selector and then press down on the top of the Selector.

- **`Eject`** safely unmounts an attached USB drive. The display will notify you when it is safe to unplug the USB drive.
- **`Reload`** rescans the **`Patches`** folder of the attached USB drive and then refreshes Organelle's patches menu. (This also unloads the current patch so note that all sound will stop until you load another patch.) Any time that you connect a USB drive to Organelle while the unit is already powered up, you should run this function. Once the reload process has completed, you will be left in the patches menu in order to load an available patch.
- **`Info`** displays the *system information screen*, which presents four pieces of information. (To exit this screen, turn the Selector left or right.)

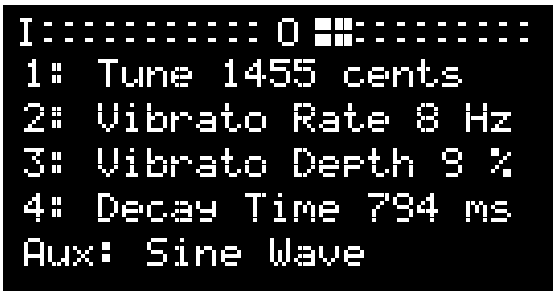


  - **Version** represents the version number of Organelle's operating system.
  - **MIDI** will list any recognized MIDI controllers that are connected. When no controllers are connected, nothing will be shown.
  - **USB drive** presents the ID number of the attached USB drive. When no drive is connected, nothing will be shown.
  - **CPU** reports the current usage that the microcomputer's CPU is experiencing. This value is regularly updated.

- **Shutdown** safely prepares Organelle to be powered off. The display will notify you when it is safe to unplug the power supply from the unit. (Once the unit has been shut down, you must remove and reconnect power to start using it again.)

Patch Information Screen

Once a patch is loaded, you will be taken to the *patch information screen*, which serves as your main performance interface.



Once a patch is loaded, Organelle treats this page as your "home screen." You can return to the menu screen by turning the Selector knob, but after a few seconds of inactivity, Organelle will automatically return to the patch information screen.

On this primary screen, each line tends to show particular information related to the current patch. (Individual patches can vary this pattern when deemed appropriate.)

- The top line provides level meters. These are representations of the audio levels that are reaching the device's **I**(nputs) and those that are ending up at its **O**(utputs). Each meter is actually a stereo representation, with the top rows showing left levels and the bottom rows showing those for the right channels.

  In the example pictured above, the audio output seems to be of similar strength on both the left and right channels. And no audio input is being received at this moment in time.

NOTE: The on-screen output meter is operating in a "post fader" mode, where the displayed signal is scaled based on the unit's current volume setting.

- The middle four lines are prefaced with **1**, **2**, **3**, and **4**, each representing that respective knob. Following the knob number is usually a short name for the parameter being altered and then a numeric representation of that parameter's current value. So in the example shown above, Knob 2 is controlling a parameter named **Vibrato Rate** that is current set to **8 Hz**.
- The bottom line is prefaced with **Aux** as it tends to provide more information about the assignment and/or current state of the Aux button. In the example above, this patch seems to toggle the waveform being used an oscillator, with a **Sine Wave** currently in use.

Organelle's Default MIDI Setup

The way Organelle handles MIDI will be relevant to all users, even if you are mainly pressing the unit's own maple keys to trigger note messages. There are certain default MIDI assignments in Organelle patches that you should know.

NOTE: In this section, terms specific to the MIDI protocol will have quotation marks around them. As this jargon predates Organelle, we would suggest consulting a general resource on MIDI if the terms aren't clear to you.

NOTE: These MIDI assignments are labeled "defaults" as you can override them in your own patches. For more on making patches and overriding the default MIDI assignments, see chapter five.

Outgoing MIDI

With the exception of the Selector knob and the Volume knob, all of Organelle's other interface elements send out MIDI messages when they are used. All MIDI messages are sent on channel one.

NOTE: All outgoing MIDI messages are sent over USB. They will be received by any connected USB MIDI device or interface that is actively listening for them.

The Keys

The 24 keys (not including the Aux button on the far left) transmit "note on" messages. The leftmost key uses note number **`60`** ("middle C," or "C3" in most MIDI systems), and the rightmost key uses note number **`83`** ("B4"), with all keys in between following this scheme.

When a key is pressed down, a "velocity" of **`100`** is transmitted. When a key is released, a "velocity" of **`0`** (zero) is sent.

The Knobs

Knobs 1, 2, 3, and 4 transmit "control change" messages using controller numbers **`21`**, **`22`**, **`23`**, and **`24`**, respectively. The full range of controller values (from **`0`** to **`127`**) is utilized.

The Aux Button

The Aux button transmits momentary "control change" messages using controller number **`25`**. When the button is pressed down, a controller value of **`100`** is transmitted. When the button is released, a controller value of **`0`** (zero) is sent.

The Pedal Port

A pedal connected to Organelle's **`Pedal`** port transmits two sets of "control change" messages.

Controller number **`64`** transmits a controller value of **`0`** (zero) for any received signal below **`64`**, and a controller value of **`127`** is sent for any received signal of **`64`** or above. This discrete, threshold behavior is particularly good for sustain-/damper-style pedals.

Controller number **26** transmits continuous values. While the general range would be from **0** (zero) to **127**, the exact range of values may vary based on the pedal connected. This continuous behavior is ideal for an expression-type pedal.

No matter what type of pedal is connected, both of these control change messages will be transmitted, and there is nothing stopping you from using both sets of messages.

Incoming MIDI Messages

While all MIDI messages received by Organelle (via USB) will be passed on to the patch that is currently loaded, certain messages are handled uniformly across all patches.

In general, the MIDI messages that are output by Organelle (as outlined in the immediately previous section) are identical to the incoming messages recognized by Organelle. This can be helpful, for example, if you want to record the movement of Organelle's controls into a sequencer as automation data. In other words, the mappings are a bit of a mirror. So let's take particular note of how incoming MIDI messages interact with and can sometimes override Organelle's on-board controls.

And similar to the outgoing messages, incoming messages should be sent on channel one.

Note On Messages

Incoming note messages can happen concurrently with note messages created by playing Organelle's keys. If incoming and internal notes are occurring at the same time, these two streams are essentially merged into one large set of notes.

Control Change Messages

Incoming "control change" messages using controller numbers **21**, **22**, **23**, and **24** replace the current values set by Knobs 1, 2, 3, and 4, respectively.

To restore a knob's control: simply turn the knob enough to register a new value. In the same way that "control change" messages are designed to work, the dominant message is always the last one received.

Incoming "control change" messages using controller number **25** affect the internal status of the Aux button. A controller value between **64** and **127** simulates the Aux button being pressed down, while a controller value between **0** (zero) and **63** simulates a release of the Aux button.

NOTE: A momentary control source, such as a damper pedal or button, would work well with this sort of threshold behavior. In certain situations, controlling the Aux button from a external "sustain" pedal could be quite effective.

Incoming "control change" messages using controller number **26** replace the current value used by Organelle for an expression-style pedal. And incoming messages using controller number **64** replace the current value used by Organelle for a sustain-/damper-style pedal. (This subtle distinction really only matters if you are making your own patches.) Similar to the knobs, using the pedal will reactive it as the current control source, updating both controllers **26** and **64**.

Program Change Messages

Incoming "program change" messages are used to change Organelle's current patch. "Program numbers" are assigned based on each patch's position in alphabetical order. Let's unpack this a bit.

Just as they are displayed in the patches menu, all currently available patches are taken in alphabetical order, and each patch is then dynamically assigned a "program number" based on its position. So if Organelle had three patches available called **A Patch**, **B Patch**, and **C Patch**, program number **1** would call up **A Patch**, program number **2** would call up **B Patch**, and program number **3** would call up **C Patch**. (If these were the only three patches available, program change messages for programs **4** and above would do nothing.)

Other MIDI Messages

Any other MIDI message is passed directly to the current patch. If the patch is configured to handle that particular message, it will respond as configured. If the patch is not listening for that message, then nothing will happen.

Using a USB MIDI Device

Using a MIDI device with Organelle is rather painless but not "hot swappable."

1. **Connect your USB MIDI device.** As long as a USB MIDI device requires no special, proprietary driver, you need only to connect it to Organelle. This can be done via an open USB port either on Organelle itself or on a USB hub that is connected to Organelle.

2. **Load the patch you want to use. If it was already loaded, please reload it.** A newly connected USB MIDI device will not be recognized by the currently loaded patch. Reloading a patch will recognize all currently connected USB MIDI devices.

That's about it. By remembering to reload your patch and knowing what MIDI messages are understood by Organelle (see the immediately previous section) and/or those understood by the particular patch you have loaded, you should be all set to use MIDI with Organelle.

Chapter Four

Loading Patches from a Computer

Bringing additional patches from your computer to Organelle.

One of the strengths of Organelle is its depth. The included factory patches represent a good sample of what is possible with Organelle, but these are by no means the only patches you can use.

As was mentioned in an earlier chapter, organelle.io is the official repository of Organelle patches and a great place to start when looking for new sounds, options, and performance approaches. (And as it tends to do, the Internet will probably have some suggestions of its own…)

Finding and downloading Organelle-ready patches is easy enough. To actually use these patches, however, requires getting them from a computer to our properly formatted USB drive, which can then be connected to Organelle for running our newfound patches. (And yes, that computer could be running Windows, Macintosh, Linux, or some other operating system. No additional software is required; the computer is just being used to download files, possibly decompress them, and then copy their folders to the USB drive.)

In this chapter, we'll look at how to work with Organelle's USB drive on a computer. But to get there, we must begin with how that drive needs to be configured, and we'll end with a few extra notes on USB usage with Organelle.

With this information in hand, we'll be able to expand our patch library. Here's to growing our set of options.

The Required USB Drive

As has probably become clear by now, Organelle will only work properly while a USB drive is attached to it. In light of this fundamental fact, a few words on the setup of that USB drive are in order.

General Information

Patches are run directly from an attached USB drive. This could be the flash drive included with Organelle or another USB disk that is appropriately configured.

- This USB disk should be formatted with a FAT file system, often associated with MS-DOS.
- This USB drive must contain a folder called **`Patches`** at its top-level. (This name is case-sensitive.)
- For each patch to be used by Organelle, the **`Patches`** folder must contain a folder named as you would like the patch name to appear. Each patch's folder must contain a file named **`main.pd`** that contains the top-level patch itself. If any other files are required by the patch, they must also be included in the patch's folder.

Folder Structure

An example directory listing of an Organelle-friendly USB drive would start like this. You can see here at least four patches, each with a required **`main.pd`** file.

```
Patches/
      32 Oscillators/
            main.pd
      Analog Style/
            blsaw.pd
            distort.pd
            main.pd
            run.sh
            sequencer2.pd
            simple.pd
      Arpeggio - Double/
            counter-down.pd
            counter-up.pd
            counter-updown.pd
            delay2sec.pd
            info.txt
            main.pd
            master-metronome.pd
            sequencer2.pd
```

```
        Basic Poly/
              main.pd
              voice.pd
    …
```

Working with the USB Drive on a Computer

Rather than guide you through web browsing, we will assume that you have already downloaded some new patches either from organelle.io or another source.

> **NOTE:** If the patches you downloaded are ZIP files, be sure to 'decompress' those files into folders before continuing. On most modern operating systems, this can be achieve by simply opening the ZIP file from the system's file browser.

From here, we need to connect the USB drive you are using with Organelle to your computer. A couple important notes.

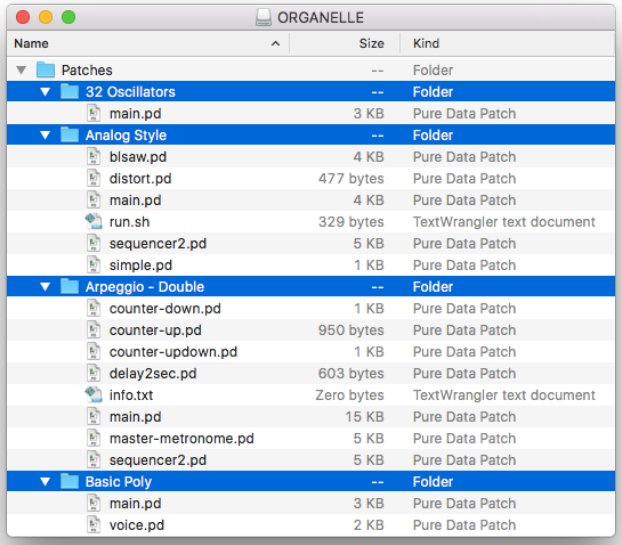
- **If the USB drive is currently connected to Organelle, properly `Eject` the disk before removing it.** This option can be found by accessing the system information screen from the system menu. (See the end of this chapter for specific instructions.)
- **When using the included USB drive, be sure it is flipped the right way before connecting to your computer.** The drive that comes with Organelle has an extremely low profile, consisting of the bottom half of most standard USB connectors/cables. As such, it is possible to plug the drive in upside down. This is generally harmless but better to be avoided.

  Be sure that the exposed pins — I think of them as teeth — are facing "up," toward wherever the top of a connected USB cable would go. (On Mac laptops, for example, the pins should face upward, toward the sky.)

Once the drive is connected and seen by your computer, we can proceed.

Folder Structure Revisited

Earlier in this chapter, we went over the required folder structure for a usable USB drive. Now that the drive is connected to a computer, let's see the same structure in a more familiar, graphical view.



The top-level of my USB drive, whose disk name is **`ORGANELLE`**, is shown above. Inside of the top-level **`Patches`** folder, the folders for the first four patches are selected (with blue coloration), and their contents are exposed.

Again, the requirement here is that each folder contains a **`main.pd`** file that serves as the primary file for that patch. As long as this file is in place, that patch will appear in Organelle's patches menu as the folder name. So in the example shown above, the first four patches shown on Organelle would be **`32 Oscillators`**, **`Analog Style`**, **`Arpeggio - Double`**, and **`Basic Poly`**.

Making Changes to the USB Drive

Assuming your USB drive is formatted correctly and the **`Patches`** folder is appropriately named and located, making changes to your available patches is as simple as working with files on your computer.

To add a patch to your Organelle's USB drive: copy the properly formatted patch folder into the USB drive's **`Patches`** folder.

To backup a patch: copy the patch's folder to a location on your computer.

To rename a patch: rename the patch's folder, just as you would rename any folder on your computer.

To delete a patch: delete the patch's folder, just as you would delete any folder on your computer.

Taken together, these basic functions make it easy for you to organize and sort your patches. Another useful idea is to prepare a set of patches for a live performance by doing something like this.

1. Back up all of your patches from the USB drive onto your computer (a good general starting point).
2. Copy any new patches you will need from your computer to the USB drive.
3. Delete any patches off of the USB drive that you will not need for this set.
4. Sort the remaining patches by adding a number to the front of each patch name (e.g., if **`Basic Poly`** was the first patch needed, I would rename it something like **`1-Basic Poly`**.) This allows you to order the patches you will need while preserving their names.

Additional USB Information

Here are couple other procedures and facts to bearing in mind while working with USB on Organelle.

- **To safely unmount a connected USB disk:** from the patch information screen, turn the Selector knob to call up the menu screen. Turn the Selector to the left (upward) to move to the menu screen's **`SYSTEM`** section, and select and engage the **`Eject`** function. The display will notify you when it is safe to unplug your USB drive.
- **USB devices are not immediately hot-swappable.** While Organelle does not need to be power cycled when a USB device is removed, any device removal will trigger Organelle to quit the running instance of Pure Data and close out the current patch. The display will notify you when this has happened (including after you disconnect a safely ejected USB drive). You can then use the Selector to load or reload your patch from the menu screen's patches section.

Chapter Five

Preparing Patches for Organelle™

Making Pure Data patches Organelle-friendly.

As was discussed back in chapter one, the patches loaded by Organelle are patches for *Pure Data* (*Pd*), a visual multimedia programming environment. Whenever a patch is loaded by Organelle, its internal microcomputer actually runs the Pure Data application, which renders the patch in realtime and sends any resultant audio to the physical audio ports on the back of Organelle.

A large part of what is happening behind the scenes is a handshake of data between Organelle and Pure Data. Organelle sends all control signals, such as incoming MIDI messages, audio input signals, and hardware status information (current knob positions, pedal port levels, etc.) to Pure Data. And in return, the current patch works with Pure Data to return text information to be shown on Organelle's display, the status/color to be shown on the Aux button's LED, and, of course, the rendered audio that is delivered to Organelle's outlets.

While this might sound a bit dense, we have created a system that makes adapting and/or creating Pure Data patches for Organelle as simple as possible. This allows your patches to keep working in their normal fashion, with you lightly amending them to receive and send messages to Organelle along predefined routes. This chapter will outline this system after some prefatory remarks on Pure Data itself.

Teaching you how to use Pure Data is beyond the scope of this chapter (and this manual); learning it is something that you will need to do on your own. In truth, patching in Pure Data is far more approachable than you probably think. If you are on the fence about learning to patch or not, a little exploration is almost certainly worth your time.

Ladies and gentlemen, this is now a diving pool, and by reading this chapter, you are preparing yourself for the leap. Goggles on? Floaties secure? Okay, let's do this.

A Little Bit about Pure Data (Pd)

While this chapter will not teach you how to use Pure Data, you should have a basic understanding of what it is. A few introductory remarks are in order.

What is Pd? And where?

Pure Data (*Pd*) was created by Miller Puckette and has been available in some form since 1996. Various versions/distributions of the program are currently available. The one used by Organelle is the official, basic version, so-called *Pd vanilla* (or just plain *Pure Data* [*Pd*]).

Pure Data is free, and it is available for Mac, Windows, and Linux platforms, etc. To continue with this chapter, you should at least open Pure Data on your computer. Current builds for various operating systems can be downloaded from either the official site or from Miller Puckette's site.

"Programming" in Pure Data

Pure Data is a visual multimedia programming environment. What this means in that "programming" here consists of selecting *objects* (think *modules*) that have been provided and then interconnecting them with *cords* (yes, like *patch cords*). The clear analogy here is to modular sound synthesis, and many of the same concepts apply in programming Pure Data.

Since the low-level objects are already built for us, our task is to think and work at a higher level, creating a patch that achieves our desired objective. This higher-level approach is the key to Pure Data: you focus on concepts and functionality, trusting that the fundamental blocks will do their job (when connected appropriately).

A brief aside. Pure Data can be stable, and it can also be unstable. The difference is almost always the quality of the programming. While this sounds like a great deal of responsibility, realize that environments like Pd are really software sandboxes; you are encouraged to play around because you are rather unlikely to break something beyond repair. The worst case scenario is that you waste some time, but even then, you will definitely learn something!

Learning More on Your Own

This chapter will continue with some details about the operation and objects of Pure Data from the perspective of Organelle, but this short preface is as far as we will go with general Pure Data concepts. The Pure Data homepage has a number of resources, including

manuals in various languages. If you want to drop straight into the primary manual, you can do that directly from Miller Puckette's homepage as well.

As for resources beyond manuals, a few links and tutorials are gathered on the Pd homepage. Beyond that, a web search will trawl endless other options.

Preparing a Patch for Organelle

To start working with Pure Data patches, there are two approaches you can take.

- **Find generic Pure Data patches online and edit them for use on Organelle.** This approach has a gentler learning curve, allowing you to see working patches in action before attempting to create your own from scratch. As such, newcomers to Pure Data and those who are on the fence about programming should begin here.
- **Create patches of your own.** Not for the faint of heart, this approach is more suitable for those who have some familiarity Pure Data (or a similar dataflow/patching programming language) already. But if you get inspiration from staring at a blank canvas and reading a lot of documentation, you can try this way too.

Regardless of which method you pick, you can always switch to the other one at any time. And the following section will serve you equally well, whether you are aiming to tweak or rolling your own. Shall we?

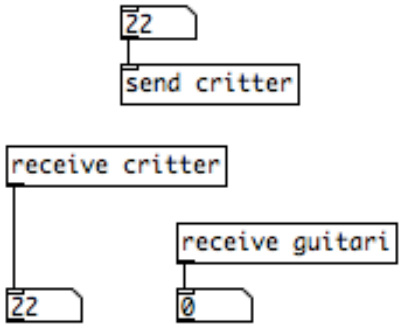
How Organelle and Pure Data Communicate

As the introduction to this chapter set out, Organelle's microcomputer runs its own copy of Pure Data, and the Organelle hardware and Pure Data application are constantly passing data back and forth. This means that any patch you are working on should handle the tasks of receiving data from and sending data to the Organelle hardware.

Remote Messaging

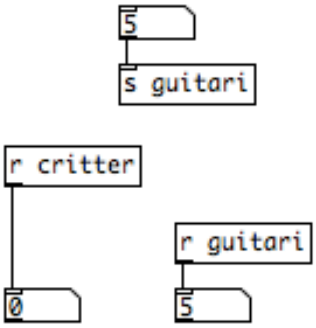
While messages and signals are intended to travel across patch cords in Pure Data, Pd also provides another option. Messages can be sent *remotely* by means of a set of special objects.

Pure Data's **send** object allows the direct transmission of messages to a designated **receive** object. As long as both objects are given the same name (and you can use any word for the name), they are connected across the same remote buss and will share their messages.



The small patch example above shows a **send** object, whose first argument, **critter**, sets the remote buss where incoming messages are being sent. When that **send** object receives the number **22**, it is passed on to all **receive** object who are similarly set with the argument **critter**. And as we can see in the above example, the **[receive critter]** object did indeed receive **22** while **[receive guitari]** did not.

As the **send** and **receive** objects may get a lot of use, you can also abbreviate these objects as **s** and **r**, respectively.



In this abbreviated example, a number 5 is received by **[s guitari]**, and only the **[r guitari]** object received that **5**. This is exactly how things should work.

Logistically speaking, you can have as many **send** and **receive** objects as you like, and any number of them can be set to the same remote buss. Messages are easy like that.

You can also send signals remotely, but the situation is a little more complicated as there are two methods for doing so.

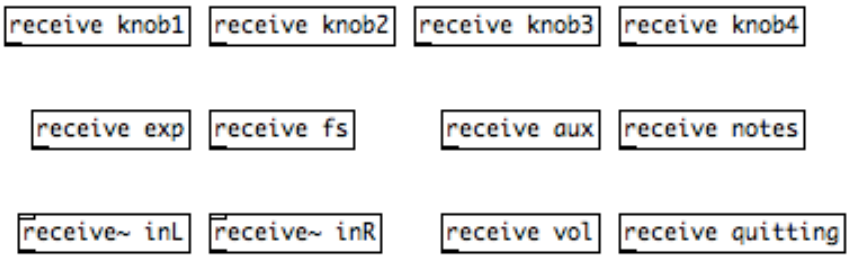
- The **send~** and **receive~** objects transmit signals remotely. They work similarly to the message-based **send**/**receive** objects, and they can even be abbreviated as **s~** and **r~**. The difference is that only one **send~** (or **s~**) can use a particular buss name, but an endless number of **receive~** objects can be tuned to that channel.
- The **throw~** object can also transmit remote signals that are then received by a **catch~** object of the same name. These objects are essentially the opposites of **send~** and **receive~**, allowing multiple **throw~** objects to share the same buss name but only one **catch~** object can listen to that feed.

For most of you, it is helpful enough to know that seeing **send~**/**receive~** or **throw~**/**catch~** objects means that signals are being passed remotely, just as **send**/**receive** objects indicate that messages are flowing over-the-air.

Finally, we have discussed the obvious feature of remote transmissions in Pure Data (no patch cords!). But there is another benefit to using these methods: the transmitter (**send**, **send~**, **throw~**) and receiver (**receive**, **receive~**, **catch~**) objects don't need to be in the same patch. This is at the heart of how Organelle and Pure Data work together.

Reserved Remote Busses for Organelle

Organelle transmits the data that your Pure Data patch will need across specified, named remote busses. This means that configuring our patches for Organelle simply means handling these incoming (and outgoing) data streams.



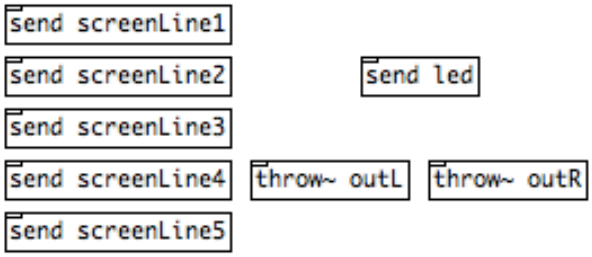
The above image shows the various paths that Organelle may use to transmit data to the current patch. These various **receive** and **receive~** objects can be placed in your patch to fully support Organelle. A little information about each follows.

- **[receive knob1]** – Incoming messages from the hardware's Knob 1; floating point numbers from **0** (zero) to **1**

- **`[receive knob2]`** – Incoming messages from the hardware's Knob 2; floating point numbers from **`0`** (zero) to **`1`**
- **`[receive knob3]`** – Incoming messages from the hardware's Knob 3; floating point numbers from **`0`** (zero) to **`1`**
- **`[receive knob4]`** – Incoming messages from the hardware's Knob 4; floating point numbers from **`0`** (zero) to **`1`**
- **`[receive exp]`** – Incoming messages from the hardware's Pedal port; a continuous range of floating point numbers, between **`0`** (zero) to **`1`**
- **`[receive fs]`** – Incoming messages from the hardware's Pedal port; a boolean value of either **`0`** (zero) for pedal released, or **`1`** for pedal depressed
- **`[receive aux]`** – Incoming messages from the hardware's Aux button; a boolean value of either **`0`** (zero) for button released, or **`1`** for button depressed
- **`[receive notes]`** – Incoming messages from the hardware's keys; each message is a list comprising the key number (**`60`**–**`83`**, depending on which key is sending the message) and the velocity (**`100`** for a key being pressed, **`0`** for a key being released)
- **`[receive~ inL]`** – Incoming signal from the hardware's left audio input; audio signal represented in the standard range of **`-1`** to **`+1`**
- **`[receive~ inR]`** – Incoming signal from the hardware's right audio input; audio signal represented in the standard range of **`-1`** to **`+1`**
- **`[receive vol]`** – Incoming messages from hardware's Volume knob; floating point numbers from **`0`** (zero) to **`1`**
- **`[receive quitting]`** – Incoming message from Organelle whenever the current patch is being unloaded; a **`bang`** message that can be used as a trigger (e.g., to close an audio file being written or to send "note off" messages)

NOTE: The **`bang`** message sent over to a **`[receive quitting]`** object only comes about 100 milliseconds before the patch is actually closed out.

And then there are the messages that Organelle will respond to. Again, the following forms are for use in your patches so that messages can be transmitted back to Organelle.



- **`[send screenLine1]`** – Outgoing messages that set the first line of Organelle's display after the level meters (usually, this line displays the status of Knob 1); any message type will be accepted (including list and symbol)
- **`[send screenLine2]`** – Outgoing messages that set the second line of Organelle's display after the level meters (usually, this line displays the status of Knob 2); any message type will be accepted (including list and symbol)
- **`[send screenLine3]`** – Outgoing messages that set the third line of Organelle's display after the level meters (usually, this line displays the status of Knob 3); any message type will be accepted (including list and symbol)
- **`[send screenLine4]`** – Outgoing messages that set the fourth line of Organelle's display after the level meters (usually, this line displays the status of Knob 4); any message type will be accepted (including list and symbol)
- **`[send screenLine5]`** – Outgoing messages that set the fifth line of Organelle's display after the level meters (usually, this line displays the status of the Aux button and any function under its control); any message type will be accepted (including list and symbol)
- **`[send led]`** – Outgoing messages that set the status of the Aux button's LED; **`0`** (zero) triggers off, **`1`** triggers red, **`2`** triggers yellow, **`3`** triggers green, **`4`** triggers teal, **`5`** triggers blue, **`6`** triggers purple, **`7`** triggers white
- **`[throw~ outL]`** – Outgoing signal that will be passed to the hardware's left audio output; should be an audio signal in the standard range of **`-1`** to **`+1`**
- **`[throw~ outR]`** – Outgoing signal that will be passed to the hardware's right audio output; should be an audio signal in the standard range of **`-1`** to **`+1`**

A patch that is prepared to exchange these various messages/signals with Organelle is "fully compatible," but there is no requirement that you use or even include all of these

objects. As with most facets of Organelle, the choice of what to use — and what to ignore — is completely yours.

Overriding Standard MIDI Behavior

We just finished saying that you can generally change how Organelle works so here is the proof. In case you want to disable the MIDI messages used by Organelle, we have made it easy to do so. (If you are uncertain why you would want to do this, feel free to ignore this section.)

There are two additional data streams for asking Organelle to do this.

- **`[send midiOutGate]`** – Outgoing message to Organelle; a boolean value of either **`0`** (zero) for disabling normal MIDI out messages, or **`1`** for restoring them
- **`[send midiInGate]`** – Outgoing message to Organelle; a boolean value of either **`0`** (zero) for disabling normal MIDI in messages, or **`1`** for restoring them

By disabling either (or both) of these functions, you are canceling the standard MIDI messages used by Organelle (see chapter three). The assumption is that your patch will then provide its own methods for dealing with MIDI. But again, it's your prerogative.

A Few Tips on Patch-building

Here are a few thoughts and suggested guidelines for preparing and/or making Pure Data patches for Organelle.

- **Build the simple version first.** Particularly if this kind of programming is new to you, the best thing you can do is start by building something simple that works. And then slowly expand it, adding or changing only one function at a time.
- **Break your idea into pieces.** If I were, for example, building a basic playback sequencer, I would break this down into three conceptual pieces: a clock source (like a metronome), a data structure (for storing the notes I want played back), and a labeling function (to convert the clock source's nameless "ticks" into the "1, 2, 3…" etc. that would tell the data structure which address to playback at this moment). Some of these pieces might consist of single objects. Others would require small programming blocks.

- **Try to keep CPU usage under 75%.** Even the most carefully constructed patches might require additional resources in a performance situation. While only a guideline, this is a practice that we ourselves endeavor to follow.

NOTE: Current CPU usage can be viewed from the system information screen (see chapter three).

- **Building patches directly on Organelle is possible.** This requires setting up Organelle as a computer, with a monitor, keyboard, and mouse (see chapter six). This is just as valid as working off your personal computer. As with all workflow decisions, the choice is yours.
- **Save versions of everything you do.** Especially when making big changes, don't simply overwrite the previous version of your patch; keep a working version that you can go back to. Just in case.

Things Unsaid

In summary, we have covered all the messages that Organelle is either providing to or expecting from your patch. By equipping your patch to use those remote busses, you are enabling your patch to be Organelle-friendly and live well in this ecosystem.

But we have not told you how to make a patch. And again, this is by design. Aside from being well beyond the scope of this document, using Pure Data and creating something "useful" is truly an example of beauty being in the eye of the beholder.

Looking at the factory patches will give you a sense of certain possibilities, and studying the Pure Data resources listed at the top of this chapter will school you in general methods of working. If you do choose "to code," the path forward is yours. Our only overarching advice is to build something that you would like use, and then put one foot in front of the other.

Chapter Six

Attaching an HDMI® Display

Using Organelle as a computer.

Well, you have made it to the final chapter. Mazel tov! We hope you have enjoyed the journey thus far, and it is worth mentioning that repeat visits to this document tend to be rewarding.

This chapter deals with Organelle's HDMI port, which has been sitting there the whole time. The reason for our longstanding neglect is simple: only users who want to tweak/program their own patches might benefit from connecting Organelle to a monitor, television, or what have you. So if you are interested this chapter but have somehow skipped chapter five, you may want to reconsider that.

While we have continually mentioned the microcomputer within Organelle, we have also pointed out the "un-computer" nature of this instrument. But we also recognize that you might want run Organelle with a monitor, keyboard, and mouse — some folks might find it particularly nice to edit patches in this fashion. So our main theme has reemerged: it's your choice.

This chapter will walk us through the general use of Organelle along with HDMI, which presumes the use of a keyboard and mouse as well. We will do this by going through a general narrative of using Organelle as a computer.

Let's get visual. And peripheral.

Keyboards and Mice

Connecting an HDMI monitor to Organelle is simple enough. But the function of the HDMI device is to provide the visual output of Organelle's functioning microcomputer. (As we will see very shortly, connecting only a monitor will show you, well, not much.) As with any computer, you need a keyboard and mouse actually interface with Organelle.

Just about any USB mouse should work with Organelle, and most PC-style USB keyboards should also be fine. Additionally, mice and keyboards that have their own USB wireless

dongles should also work with Organelle. So long as the data is coming across a USB port, your peripherals will probably work.

> **NOTE:** While we have aimed to support regular USB keyboards, not all manufacturers implement the general USB standards in the same way. Accordingly, some keyboards may not work with Organelle. Please report any finding of incompatibility on our forum.

Finally, we are talking about more and more USB devices being used with Organelle. (And don't forget the USB drive housing your **`Patches`** folder!) Before you worry about running out of ports, remember that a USB hub can be connected to Organelle.

Operating Organelle as a Computer

After you have connected an HDMI monitor and powered it on, you will immediately see a semi-cryptic terminal window for text entry. To optimize performance, Organelle runs in this fashion (with no graphical user interface, or GUI) by default.

This window helpfully suggests that you run the command **`startx`** to start up the standard graphical environment. So if we want to go beyond this screen, we will need a keyboard. (And we can say now that if you want to edit your Pure Data patches on Organelle, you will definitely need a mouse as well.)

To start Organelle's graphical operation mode: type **`startx`**, and then press the [ENTER]/[RETURN] key to execute the function.

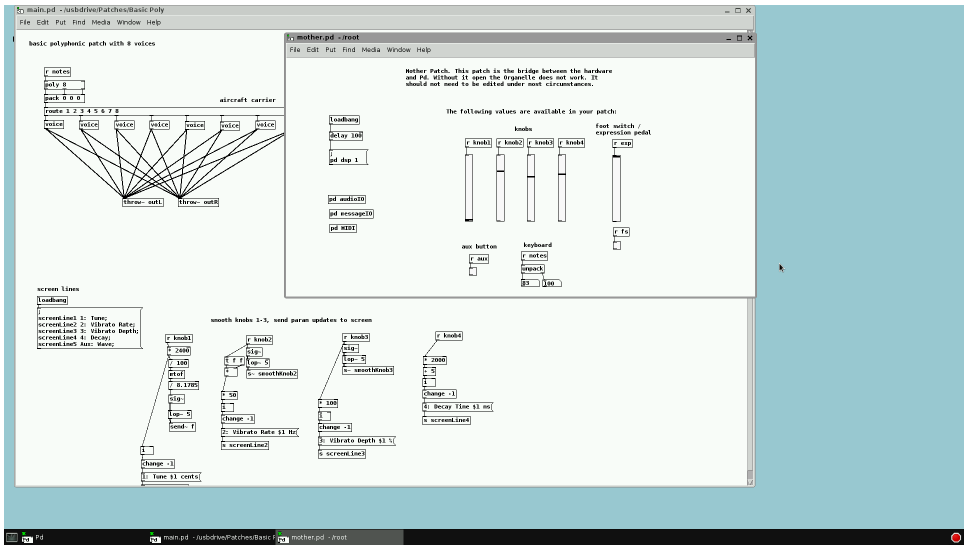
> **NOTE:** Booting Organelle's graphical operation mode causes the system itself to be reloaded. This means that any currently loaded patch will be unloaded, and any audio output being produced will cease.

The options on this screen are fairly sparse. This matches the character of the running operating system, which has been stripped down in favor of achieving the most stable audio performance. There are four elements here.

- The disk icon labeled **`USB Drive`** at the top left of the screen represents the USB drive containing your **`Patches`** folder.
- The **`Read me First!`** file contains some helpful notes from our technical team. You should probably do as they say.
- The dark icon at the bottom left of the screen represents a command-line interface (CLI). If you click on this icon, a terminal emulator instance is started.
- The red octagonal icon at the bottom right of the screen represents a stop sign. When you are done working in this graphical operation mode, you should click this icon.

To exit Organelle's graphical operation mode: click the red octagonal icon at the bottom right of the screen. This will close out all open windows and unload the current Organelle patch, interrupting any ongoing audio output (the same as when we entered this graphical mode). This red "exit" button is the primary interface option that you will need.

While you can navigate the file system with the keyboard and mouse, the best way to load a patch is to do it from Organelle's hardware. By using the Selector encoder to choose and load a patch, you will then see the patch visually loaded by Organelle along with a crucial helper patch.



The Organelle unit itself is now functioning as we would normally expect it to: the patch has been loaded, Organelle's hardware display has shifted to the patch information screen, and audio can now be produced.

Within the computer interface, we are now seeing the behind-the-scenes implications of loading a patch. Our patch (in this case, **Basic Poly**) has been loaded, and its **`main.pd`** file is taking up most of the screen. But sitting atop the patch we expected is one we did not.

The mother.pd *Helper Patch*

`mother.pd` exists at the root (or top) directory of Organelle, which is located on the microSD that comes preloaded within the Organelle hardware. This helper patch is the other half of the data handshake between the Pure Data patches we run and Organelle's hardware.

In short, this helper patch is executing the raw communications with the Organelle hardware. (This is done using the *Open Sound Control* [*OSC*] protocol.) The **`mother.pd`** patch then acts as the data router, converting the raw data streams into the simpler remote buss communications that we covered in chapter five.

Accordingly, **`mother.pd`** is necessary for the general operation of Organelle. That is why this patch is loaded concurrently with any patch that you call up.

NOTE: In general, you should not edit **`mother.pd`**.

That being said, Organelle will use any file named **`mother.pd`** that it finds within the **`Patches`** folder of your USB drive. By copying the root directory's **`mother.pd`** to your **`Patches`** folder, you could experiment with editing this patch while keeping the master version clean. Again, you probably don't want to do this.

The Patch Load Sequence

To better understand the full relationship at hand, let's take a step back for a moment. Anytime a patch is loaded, Organelle goes through a sequence of steps.

1. If a patch is currently loaded, it receives a **`bang`** message over the **`quitting`** remote buss. This allows any "cleanup" processes to be executed.
2. If a patch is currently loaded, it then prompts the Pure Data application to quit. This effectively closes any and all open patches, including the **`mother.pd`** helper patch.
3. The Pure Data application is relaunched, and the patch we have requested is then opened.
4. The **`mother.pd`** helper patch is loaded.

Once this sequence completes, all assets needed for your patch to communicate with Organelle will be loaded and ready to go. So the general flurry of windows closing and opening that you see in Organelle's graphical operation mode is both expected and proper.

By clicking on your loaded patch, it will move **`mother.pd`** to the background (without closing it) and allow you to focus on working with your patch.

Other Operational Tips

Just a few parting notes on this mode of operation.

- **The Linux file browser can be used as you would *File Explorer* (on Windows) or *Finder* (on Mac).** If you are running Organelle as a computer, you can navigate to your USB drive in order to rename or delete files in the same fashion described earlier (see chapter three).

- **To create a new patch:** duplicate a simple patch in your **`Patches`** folder, rename the new folder, and then open the contained **`main.pd`** patch for editing. (You could also create your own "new patch" template for this purpose.)
- **Explore the factory patches.** In addition to finding ideas and platforms that you can build upon, you will also encounter some external objects that are not part of the Vanilla Pd distribution. To use an external in a patch of your own, copy it to your patch's folder.

NOTE: Externals that you encounter here are built for the Linux operating system that Organelle is running. If you are building patches on your own computer, these externals will only work if you are also running Linux (these compiled externals are platform-specific).

- **Do click the red "exit" button when you are done working in this graphical operation mode.** While you could simply unplug your HDMI monitor and resume working with Organelle as usual, this will leave some processing resources allocated for graphics that you are no longer using. It is better to revert Organelle to its normal CLI mode and keep the processor focused on audio tasks.

Appendix A

Factory Patch Listing

Analog Style

Categories: synth, sequencer

A classic sawtooth plus resonant filter patch. It features the usual controls: filter cutoff, resonance, and oscillator tuning. The Aux button controls a note sequencer: hold it down to enable recording, begin playing to start a recording, and press Aux again to finish. Then tapping Aux will toggle the sequence to start and stop.

Arpeggio Sampler

Categories: sampler, arp

Record a sample by pressing the Aux button. This audio will then be transposed across all keys of the keyboard. The knobs control: tempo, amplitude envelope for the sample, and a selection of eight different arpeggio patterns. A foot switch (via the Pedal port) can be used to turn on "latch mode" for hands-free operation.

Arpeggio Synth

Categories: synthesizer, arp

Similar to the **Arpeggio Sampler**, but with a synthesizer for the audio engine. The knobs control: tempo, a selection of arpeggio patterns, and the envelope and tone of the synthesizer. The Aux button (or a foot switch) turns on "latch mode" for hands-free operation.

Basic Poly

Categories: synth

A simple multi-waveform synthesizer. The Aux button selects from four basic waveforms: sine, square, saw, and triangle. The knobs control overall tuning, vibrato, and envelope.

Basic Sampler

Categories: sampler

Press and hold the Aux button (or a foot switch) to record a sound (two seconds maximum). Playing the keys then plays back the sound at different pitches. Use the knobs to adjust overall speed and to add an envelope to the sample, allowing you to shape the sound in interesting ways.

Metronome

Categories: utility

A basic metronome patch. Start and stop the metronome with the Aux button. Control tempo and select a beat accent (or none). The keyboard plays a basic synthesizer tone that can be useful for tuning up.

Nice Surprises

Categories: synth, sequencer

Control three square wave oscillators with this patch. The keyboard controls the base frequency of one oscillator, which is also tunable from Knob 2. The second oscillator plays a selectable harmonic of the base frequency. The third oscillator plays a variable amount of random junk frequencies at the beginning of each note.

You can also adjust how the oscillators to glide to their next frequency (the "Slidiness" factor). The Aux button controls a simple sequencer: hold it down to enable recording, begin playing to start recording, and press Aux again to finish.

Quad Delay

Categories: effect, delay

The audio input is fed through four delay lines, each with a delay time that is a multiple of a base delay time. The base delay time is set by the keyboard. This is a nice way to build up a deranged stack of sound. Knobs control feedback.

Sampler Style

Category: sampler, sequencer

Each key plays a different sample. The bottom keys are mostly drum sounds, and the top keys are mostly sound effects. It has controls for: overall reverb, playback speed, and sample decay. The Aux button controls a sequencer: hold it down to enable recording, begin playing to start a recording, and press Aux again to finish.

Sine Surprises

Categories: synth, sequencer

This is the same three oscillator synthesizer and sequencer engine as **Nice Surprises**, but here we are using sine waves instead of squares. This allows for some cool, weird organ tones. It is also four-voice polyphonic.

Tuned Delay

Category: synthesizer, delay, sequencer

A percussive impulse sample is fed through a delay line. The delay line is tuned to the keys, allowing you to pitch the sound. The effect is similar to plucked string algorithms (think Karplus–Strong). The knobs control: overall tuning, feedback, sample playback speed, and sample selection. The Aux button controls a sequencer: hold it down to enable recording, begin playing to start a recording, and press Aux again to finish.

Wepa!

Categories: sampler

Wepa!

Appendix B

Technical Specifications

Audio Specifications

- Sampling rate: 44.1kHz; Resolution: 16 bit (both at input & output)
- 2x ¼" mono sound output jacks (left & right channels)
- ¼" stereo sound input jack
- ⅛" stereo headphone output jack
- ¼" footswitch jack

Interface

- High-contrast OLED display screen
- Four parameter knobs
- Rotary Selector knob with push button select
- Volume knob
- 25x maple key
- RGB LED

Processor Specifications

- 1GHz ARM Cortex A9 with 512 MB RAM
- Linux operating system
- Boot time: ~12 seconds

Storage

- 4GB USB Drive included; patches and any required files (audio samples, etc.) are stored here

Additional Connections

- 2x USB port (2.0 standard-A type connector). USB host for: MIDI over USB & serial over USB
- HDMI output port

Power Requirement

- 9VDC, 1.0 Amp (power supply included)

Physical Characteristics

- Size: 10.5" x 3.25" x 2.125"
- Enclosure: anodized aluminum top, ABS plastic bottom, rubber foot pads

New patches are available at organelle.io

Visit our forum at forum.critterandguitari.com