MSK 014 Gracious Host
User/Build Manual

North Coast Synthesis Ltd.
Matthew Skala

May 5, 2023

Hardware documentation for the MSK 014

Contents

General notes

This manual documents the MSK 014 Gracious Host, which is a module for use in a Eurorack modular synthesizer. The module's main function is provide an interface from USB devices, especially MIDI controllers, to the CV/gate synthesizer. As the name implies, the Gracious Host functions as an embedded host, capable of connecting devices like keyboards without needing the involvement of a separate computer. It supports multiple control modes for use in different kinds of patches, and the onboard microcontroller can be reprogrammed in the field with new or modified firmware loaded from a USB flash drive through the front-panel port.

This is one of two manuals. The other manual is specifically intended for programmers who are modifying or replacing the firmware, and is of less interest for most users.

Front panel connections

The front panel of the Gracious Host is shown in Figure 1. The fragment of music notation on the panel spells GRACE as a pun on the module name: treble clef (G clef); quarter-note rest (R); and the three notes A, C, E.

Serial bus port The port at upper left is for connecting a USB device, such as a USB-MIDI keyboard. This is an *embedded host* port, capable of supporting many USB 1.0, 1.1, and 2.0 low-speed and full-speed devices relevant to synthesizer use, but it does not support the entire USB standard with every configuration of every device. Exactly what the module will do depends on what kind of device is plugged in; it will normally detect whenever a device is inserted or removed, and go into a mode appropriate for that device, with the different modes described in subsequent chapters of this manual.

The port is intended only for standard low-power devices such as MIDI controllers. The maximum current it is *intended* to support is 100 mA (which is a "unit load" according to the relevant USB standards). Some devices may attempt to draw more, and may succeed, but this port is not intended for high-power applications like battery charging.

The Gracious Host cannot make use of a hub. The USB device must be attached directly to the port, and only one USB device can be attached at a time.

Control voltage inputs The top pair of jacks, labelled "IN," serve as control voltage inputs, which may be gate/trigger or pitch control voltages depending on the operating mode. Although the module will not be damaged by voltages between the power supply rails (±12V when the module is powered up), the useful range of voltages over which the analog-to-digital converters can work properly is from 0V to +5V.

The nominal input impedance for these jacks is 100kΩ.

Bi-colour LEDs Immediately below the input jacks is a pair of bi-colour (red and green) LEDs. These display status information with a function that depends on the operating mode. For instance, in most MIDI-controller modes, the LED on one side of the module will light when that side is playing a note.

Control voltage (analog) outputs The second pair of jacks from the top, labelled "CV," are multipurpose analog control voltage outputs with an intended operating range of 0V to +5V. These are typically used for things like pitch and velocity, again depending on the mode. In a few modes where these jacks are used for trigger or gate output, they may produce a slightly higher uncalibrated voltage like +5.5V.

Gate (digital) outputs The bottom pair of jacks, labelled "GT," produce digital control voltages (gates or triggers), with "low" at 0V to within the tolerances of op amp offsets, and "high" at approximately 9V.

Specifications

The nominal impedance for the input jacks is 100kΩ. All the outputs are driven by op amps with in-the-loop 1kΩ resistors, giving them effectively zero output

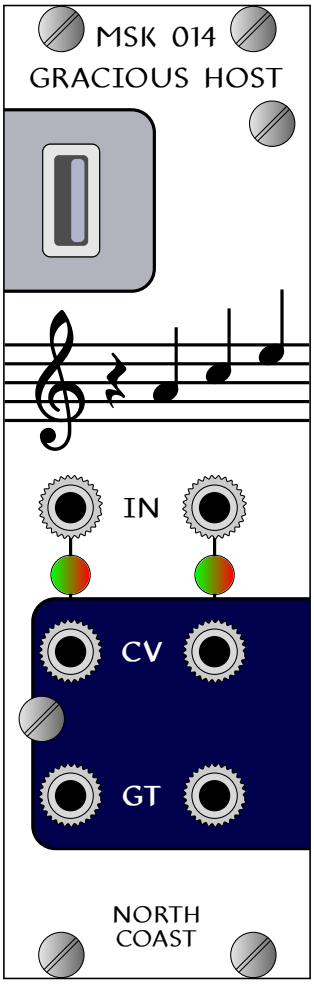


Figure 1: Module front panel.

impedance in normal use but limiting the output current to about 10 mA under short-circuit conditions.

Briefly shorting any input or output to any fixed voltage at or between the power rails, or shorting two to each other, should be harmless to the module. Note that "between the power rails" does mean it is safer to feed voltages into the module when power is applied, since without power, both power rails will be at zero. Patching the MSK 014's output into the output of another module should be harmless to the MSK 014, but doing that is not recommended because it is possible the other module may be harmed.

This module contains series reverse protection diodes and is unlikely to be damaged by a reverse power connection, but because it uses a 16-pin power connection, there are more kinds of misconnection possible than pure reversal of the main power rails. It is possible that there could be a short circuit, more likely damaging to the power supply than to the module, if the power is misconnected. The connector on the module is polarized and unlikely to be misconnected, but some caution is still recommended when connecting the cable at the bus board.

The maximum current demand of this module in normal operation is 10 mA from the +12V supply, 20 mA from the −12V supply, and a variable amount from the +5V supply. For power budgeting purposes I suggest counting the Gracious Host's +5V requirement as 130 mA, though the details are more complicated, as described below. Placing an unusually heavy load on the outputs (for instance, with so-called passive modules, output-to-output patching, or attempting to charge USB devices from the front-panel port) can increase the power supply current beyond these levels.

In more detail: the Gracious Host supplies +5V power more or less directly from the Eurorack bus to the attached USB device, so the amount drawn from the Eurorack bus will depend on what is attached to the front-panel USB port. The Gracious Host needs up to 30 mA of +5V itself, plus whatever is drawn by the USB device. USB devices, according to the relevant specifications, are supposed to draw at most 100 mA and then request more from the host if desired – but the standard Gracious Host firmware will never give the device permission to draw extra power. Thus in normal operation with standards-compliant devices, 130 mA should be the maximum consumption of +5V Eurorack power for the module and the attached device.

Now, on the one hand, nearly all devices commonly expected to be used with the Gracious Host will actually consume much less than 100 mA; 5–20 mA is more typically expected, and so in actual practice the maximum total +5V consumption is likely to be 35–50 mA. However, some high-power USB devices actually draw more than the specified 100 mA maximum without following the protocol for requesting it. The Gracious Host uses a PTC thermistor, often called a thermal fuse although it really is not a fuse at all, to protect against short circuits. This protective device has a *hold current* of 200 mA (meaning that at that level it will not trip) and a *trip current* of 400 mA (meaning that at that level it will trip in a few seconds). So, if somewhat overloaded, the module could consume as much as 230 mA of Eurorack +5V power continuously; and if heavily overloaded it could briefly draw much more before the thermal fuse trips.

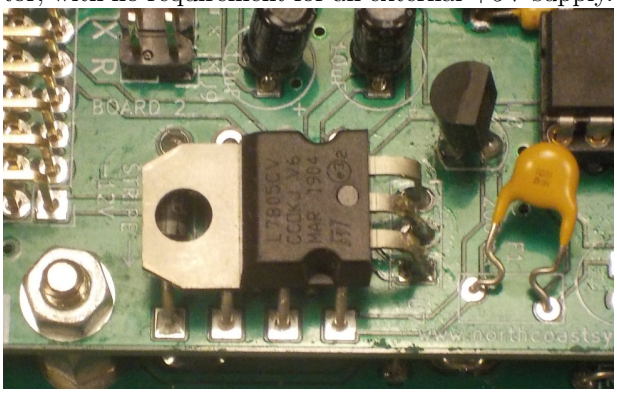
Modification: onboard +5V regulator

The Gracious Host requires +5 V power, up to 130 mA under ordinary recommended use.

In the standard configuration, this module requires a power supply that offers +5V. It will not work if plugged into a stripped-down Eurorack power supply with only ±12V. The decision was made to require +5V because at present nearly all commercial Eurorack power supplies do in fact offer +5V; the Gracious Host and the attached USB device each require a significant amount of current, with the typically spiky demand of digital circuitry; and attempting to convert ±12V to +5V on board the module would raise significant thermal, noise, and other technical issues, all just to serve the small minority of users who have no +5V supply already.

However, for special purposes such as bench-top testing, it is possible to solder in a 7805 regulator chip, on the pads next to the protection diodes, as shown. Photo depicts an early prototype with green solder mask instead of production blue, and some minor differences in other details. Then by changing the configuration jumper settings (next section), the module can be configured to draw its +5V current from the +12V instead, through the installed regula-

tor, with no requirement for an external +5V supply.

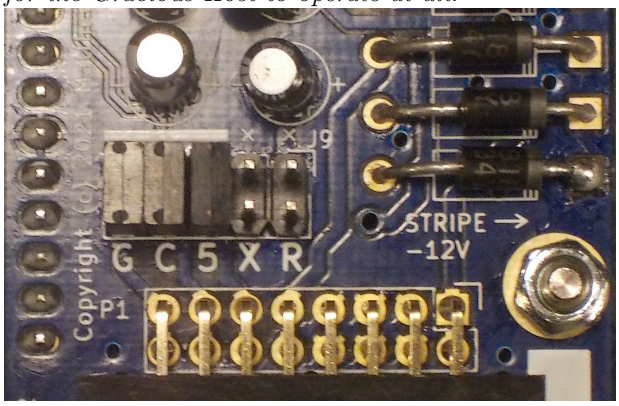


The 7805 regulator chip is not included in kits, and adding it is not a recommended or supported configuration. Depending on the amount of current drawn by the external USB device, a 7805 installed as shown without a heat sink may be liable to overheat. Using an external device that needs maximum rated power, or more, for more than a few seconds at a time may necessitate adding a heat sink to the 7805 chip. Furthermore, it is possible to set the configuration jumpers in such a way that the on-board 7805 will be cross-connected with the Eurorack bus – meaning that it will attempt to supply +5V power to other modules (possibly useful, but further increasing the likelihood of overheating), or interfere with the existing bus +5V supply if there is one after all.

The 7805, if installed, will have very little effect unless also activated with the "R" jumper described below, so it remains possible to use a modified module with bus power instead as long as you are careful with the jumper configuration.

Bus access and configuration jumpers

There is a jumper block on the back of the module for selecting the +5V source and whether to connect the first channel (left side of the front panel) output jacks to the Eurorack bus. *At least one jumper (the one selecting +5V source) must be correctly installed for the Gracious Host to operate at all.*



Left to right, the jumper positions are:

- "G": install to connect left GT output to the "gate" bus;
- "C": install to connect left CV output to the "pitch CV" bus;
- "5": install to draw +5V power from the Eurorack bus;
- "X": spare position, no effect; and
- "R": install to draw +5V power from the on-board regulator, if one is installed – otherwise no effect.

Four pins of the standard Eurorack bus are reserved for transmitting pitch and gate CV among modules. Most Eurorack cases will connect these pins among modules – usually with a separate CV/gate bus for each busboard in multi-busboard systems. Some of Doepfer's VCOs, like the A-110-1, receive pitch CV from the bus by default when no pitch CV is connected to the front panel, and North Coast's Middle Path VCO (the MSK 013) does the same. Modules like the Doepfer A-185-2 are capable of transmitting CV and gate signals onto the bus. Some Cwejman, Macbeth, MakeNoise, Malekko, MFB, Plan B, Synthwerks, TipTop, and other manufacturers' modules also use these pins in a compatible way. So with compatible VCOs, you can just plug the Gracious Host into the same busboard and have it control the VCO pitch without needing to patch them with front-panel cables. Similarly, you may be able to have the gate signal from the Gracious Host activate envelopes or VCAs without patching.

Some manufacturers have attempted to send digital signals over the CV/gate bus in a non-standard way under the name "sync bus." Despite more than one using this name, each manufacturer attempting digital use of the bus has defined its own nonstandard format for the signals (usually some variation

of MIDI adapted to voltage levels instead of current-loop), and they are not in general compatible with each other. Such use also is not compatible with the analog use of the CV/gate bus as defined by Doepfer, and the two should not be mixed.

To have the Gracious Host transmit pitch and gate CV onto the bus, install jumpers in the first two positions, labelled "G" and "C." These two jumpers have the effect of connecting the output jacks directly to the bus. It is also possible to install just one of the two jumpers, to transmit only pitch or only gate CV. The standard or default configuration for assembled modules sold by North Coast Synthesis is to install both these jumpers; but it *will* be necessary to remove them for good results if using more than one Gracious Host on the same bus. At most one module on a given Eurorack bus should be configured to drive the CV/gate bus. Setting up more than one module as the bus master is equivalent to patching their outputs into each other, and is unlikely to work well.

Install a jumper at the third position, labelled "5," to connect the Gracious Host's internal +5V supply to the Eurorack bus. This should always be done, for standard unmodified Gracious Host modules. The only reason not to install this jumper would be if the module has been modified with an onboard regulator, in which case you could install the jumper in the fifth position ("R") instead, to use the installed onboard regulator. It would also be possible to install jumpers at both positions "5" and "R," in which case the modified Gracious Host will attempt to regulate +12V power down to +5V *and also* supply that power to other modules through the bus. Doing so is potentially dangerous, but could possibly be useful in a very small system where it might be desired to support some other +5V module when the main power supply has no +5V rail.

The fourth jumper position ("X") is unconnected, and the fifth ("R") is also unconnected in a standard build. So, in an unmodified module, these two jumper positions may be used as storage locations for the two extra jumpers when CV/gate bus access is *not* desired. That way, there is less risk of losing the jumpers.

Source package

A ZIP archive containing source code for this document and for the module itself, including things like machine-readable CAD files, is available from the Web site at https://northcoastsynthesis.com/. Be aware that actually building from source requires some manual steps; Makefiles for GNU Make are provided, but you may need to manually generate PDFs from the CAD files for inclusion in the document, make Gerbers from the PCB design, manually edit the .csv bill of materials files if you change the bill of materials, and so on.

Recommended software for use with the documentation and hardware source code includes:

- GNU Make;
- LaTeX for document compilation;
- LaTeX.mk (Danjean and Legrand, not to be confused with other similarly-named LaTeX-automation tools);
- Circuit_macros (for in-document schematic diagrams);
- Kicad (electronic design automation);
- Qcad (2D drafting); and
- Perl (for the BOM-generating script).

The kicad-symbols/ subdirectory contains my customised schematic symbol and PCB footprint libraries for Kicad. Kicad doesn't normally keep dependencies like symbols inside a project directory, so on my system, these files actually live in a central directory shared by many projects. As a result, upon unpacking the ZIP file you may need to do some reconfiguration of the library paths stored inside the project files, in order to allow the symbols and footprints to be found. Also, this directory will probably contain some extra bonus symbols and footprints not actually used by this project, because it's a copy of the directory shared with other projects.

The source package also contains source code for the module firmware, which is written in PIC24 assembly language and documented in the *MSK 014 Gracious Host Programmer's Manual*. Working with that code will require additional software tools – in particular, Microchip's version of the GNU assembly language toolchain.

The whole package is covered by the GNU GPL, version 3, a copy of which is included in the file COPYING.

PCBs and physical design

The enclosed PCB design is for two boards, each $3.90'' \times 1.50''$ or 99.06mm×38.10mm. The two boards are intended to mount in a stack parallel to the Eurorack panel, held together with M3 machine screws and male-female hex standoff hardware. See Figure 2. With 12mm of clearance for the power connector and cable, the module should fit in about 39mm of depth

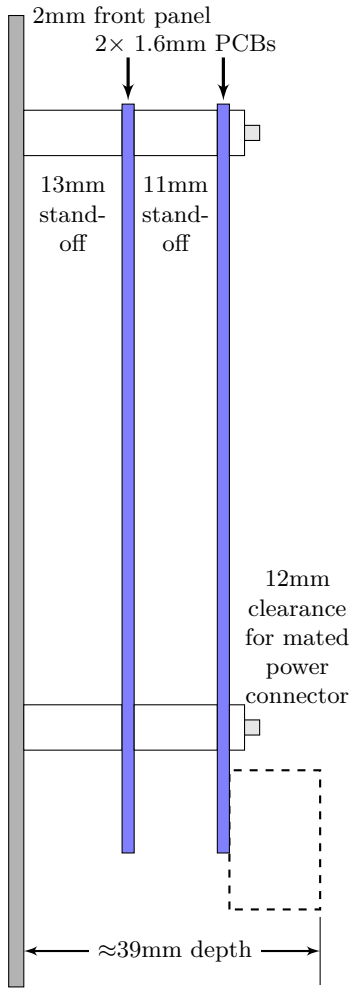


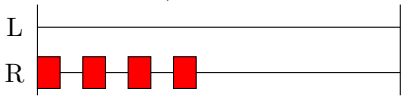
Figure 2: Assembled module, side view.

measured from the back of the front panel.

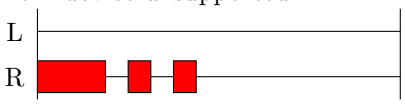
Blink codes

The two bi-colour LEDs on the front panel are used for multiple purposes. In most modes they light up in time with the notes being played. They flicker briefly when a USB device is inserted, both as a generic indicator that something is happening, and to support debugging of the stages of the USB attachment process. But in some special situations the LEDs display coded blinking patterns that report the module's status or error condition, and these patterns are illustrated in this manual with diagrams like the ones below. Time reads left to right, with the entire diagram representing a cycle that repeats every 1.05s.

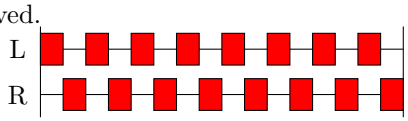
If a hub is inserted in the USB port – bearing in mind that the Gracious Host does not work with hubs – it will display bursts of four short red flashes (Morse code "H" for "hub") on the right LED, leaving the left LED dark, until the hub is removed.



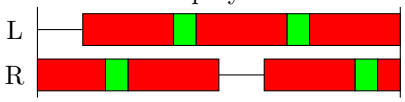
For an unsupported device other than a hub, the blink code is similar to the one for "hub," but the flashes are in a long-short-short pattern; Morse code "D" for "device unsupported."



In the event that something goes wrong inside a per-device USB driver and the module is unable to recover while the device is attached, it displays very fast back-and-forth red flashes until the device is removed.



If the module gracefully detects a trip of the USB polyfuse, indicating a short circuit or that someone tried to charge a non-compliant high-power device through the USB port, then in principle the firmware will try to display a pattern of mostly red interrupted by short green flashes and pauses. Really, though, this display is unlikely to work; in actual testing of an overloaded port condition the module is often sufficiently messed up that it shuts off until power-cycled, or at least until the polyfuse cools down.



There are other blink codes specific to the calibration procedure, which are illustrated in the section on calibration.

Use and contact information

This module design is released under the GNU GPL, version 3, a copy of which is in the source code package in the file named `COPYING`. One important consequence of the license is that if you distribute the design to others – for instance, as a built hardware device – then you are obligated to make the source code available to them at no additional charge, including any modifications you may have made to the original design. Source code for a hardware device includes without limitation such things as the machine-readable, human-editable CAD files for the circuit

boards and panels. You also are not permitted to limit others' freedoms to redistribute the design and make further modifications of their own.

I sell this and other modules, both as fully assembled products and do-it-yourself kits, from my Web storefront at http://northcoastsynthesis.com/. Your support of my business is what makes it possible for me to continue releasing module designs for free. The latest version of this document and the associated source files can be found at that Web site.

Email should be sent to
mskala@northcoastsynthesis.com.

Non-USB operation

The Gracious Host is meant to work with a USB device like a MIDI controller, but as a default, with no USB device connected, the standard firmware will function as a baby synth voice. See Figure 3 for the panel jack assignments in this mode.

The left CV input is volt per octave pitch, with a nominal range from 0V for MIDI note 36 (C two octaves below Middle C, 65.406Hz) to 5V for MIDI note 96 (C three octaves above Middle C, 2093.004Hz). The right CV input is for a gate signal (threshold approximately 1.62V). Both LEDs will light, in red, when the gate is high.

The left CV output is a semitone quantized version of the CV input. The right CV output is an ADSR envelope with fixed parameters. And the two digital outputs are square wave oscillators: unquantized pitch on the left, quantized pitch on the right.

The unquantized oscillator runs only while the gate is high, whereas the quantized oscillator runs continuously once started. The concept here is that you could use the module with an external VCA and the quantized and envelope outputs to get sound during the decay tail, or just use the unquantized output directly and have a very basic synth voice in the Gracious Host alone.

Note that the digital outputs, as usual, produce only the fixed levels of 0V for low and approximately 9V for high; so the square waves on these outputs include a DC offset of approximately 4.5V.

If you connect a USB device, the module will leave this mode and attempt to do something appropriate to the device instead.

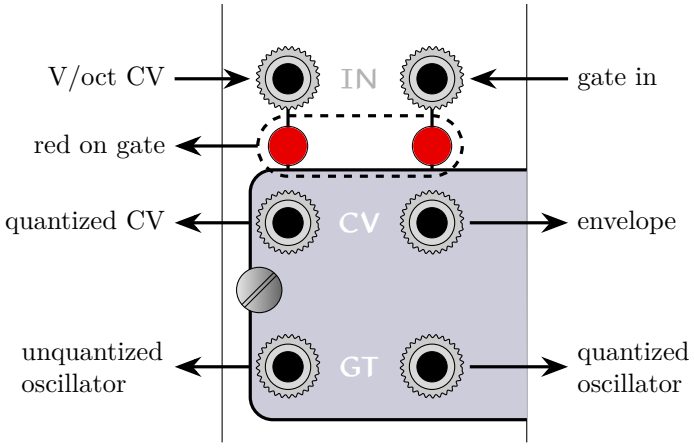


Figure 3: Jack/LED functions for non-USB mode.

MIDI interface

The Gracious Host's MIDI subsystem handles connections to USB MIDI devices like (musical) keyboards, DIN-to-USB cables, and synthesizers. The same backend also processes MIDI events generated by the typing (QWERTY) keyboard driver.

USB compatibility

The usual pattern with USB standards is that the governing organization defines a complicated protocol intended to capture every possible strange feature that an unusual device might support; and a simplified version representing the lowest common denominator. Then most implementors just use the simplified version, and the few who are building unusual devices that really require a complicated protocol, ignore the official one anyway and make up their own proprietary interfaces to use instead, which they don't document, so you can only access those devices through reverse engineering or with the manufacturers' proprietary drivers and on the operating systems they support. USB MIDI is no exception.

The Gracious Host will work with most *class compliant* USB MIDI devices in the real world. Technically, that means USB devices which expose an interface of class 1 subclass 3 and one BULK IN endpoint for MIDI stream data. Things like DIN MIDI to USB cables, controller keyboards, and so on, are likely to work. The Gracious Host is only intended to receive, not send, MIDI data from the USB port, so USB devices like synthesizers will probably work to the extent they can be used as MIDI controllers, but not as synthesizers per se. Devices that combine multiple synthesizers, audio interfaces, MIDI ports, and other things in a single product, are iffy.

Devices that are not *class compliant* will not work. In particular, a device that requires a special software driver of its own to work with a personal computer and cannot be used with a PC operating system's generic driver for that category of device, is likely not to work with the Gracious Host.

The Roland UM-ONE DIN to USB MIDI cable has a switch on it for "TAB" or "COMP." The vendor documents this switch as selecting whether you want to plug it into a "tablet" or a "computer," but what it really does is it selects class compliance: in "TAB" mode the cable is class compliant and will work with the Gracious Host, and in "COMP" mode it isn't and won't. So you should always use it in "TAB" mode with the Gracious Host. I think requiring a manual selection between the two interfaces (instead of exposing them both for automatic selection through the USB auto-configuration mechanism) is certainly a violation of the spirit, and maybe also the letter, of the USB standards on Roland's part. Other devices probably have similar issues, so if you are trying to connect some device similar in nature to a UM-ONE and find it doesn't work with the Gracious Host, try looking for any kind of compatibility mode switch or similar configuration setting, and change it.

I have an Akai MPK Mini keyboard that is basically class compliant, except it sends 64 bytes of apparently random garbage through the MIDI interface every time it boots up. The Gracious Host will ignore that burst, as well as any similar behaviour from other devices. I'm not sure Akai still sells this product anymore, and if they do, it may well be that the current version has been updated not to send bursts of garbage.

The Gracious Host does not support the new features introduced in USB MIDI 2.0. Since all USB MIDI 2.0 class compliant devices are required to also support USB MIDI 1.0 connections, the new version should not create any compatibility problems with the Gracious Host. The USB MIDI 2.0 standard does remove some of the special features of USB MIDI 1.0 for complicated devices that the Gracious Host did not support anyway (and, it's clear, almost nobody supported).

General comments on MIDI

The Gracious Host's function is usually determined by the channel of the most recent MIDI event it has received. The channels are not fully independent and, with the exception of pairs designed to work together like Channels 8 and 9, it usually will not work well to send messages to multiple channels at once. Some

data structures in the firmware are shared, so you may find for instance that changing the pitch bend on Channel 1 also has the effect of changing it on Channel 3.

The Gracious Host supports only those MIDI messages that are directly relevant to its functions as described in this documentation. Some features of MIDI not relevant to the Gracious Host, like Omni Mode, System Exclusive messages, and so on, will be ignored even if the MIDI organization has declared them mandatory.

Running Status is supported.

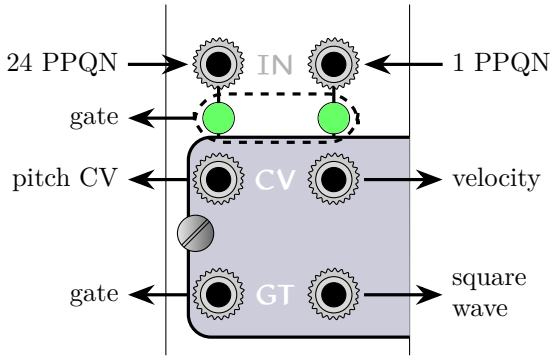
Pitch Bend messages are supported, and the range is ±2 semitones by default. Support for adjustable pitch bend range (MIDI Registered Parameter Number 0) is planned but not yet implemented.

New features of MIDI 2.0 are not supported.

The Gracious Host (regardless of channel mode) will process the MIDI Timing Clock message, which defines a 24 PPQN clock, in order to set the tempo used for things like the arpeggiator channels. This clock is also available on an output jack, or similar clocks are accepted on input jacks, in some channel modes; and when using a typing keyboard (described in the next chapter) the Scroll Lock LED blinks in time with the MIDI clock and the keypad Insert key can be used as a tap tempo input. The module will process the MIDI Start message as a reset to the start of the current quarter note, but it ignores other MIDI sequencer-control messages (Stop, Continue, Song Position Pointer, and Song Select).

Channel 1: mono CV/gate with velocity and square wave

In this mode the Gracious Host can control a single channel of a CV/gate synthesizer. Note On and Off messages translate to pitch and velocity control voltages with a nominal 0V to 5V range representing MIDI notes 36 to 96 with any relevant pitch bend, and velocity values 0 to 127. Both LEDs glow green when a note is active. The righthand gate output jack generates a square wave (low value 0V, high value 9V nominal) at the frequency corresponding to the selected MIDI note. Note that while in this mode (that is, when the most recent note played was on Channel 1), the oscillator continues running indefinitely at the frequency of the most recent note even after the note has ended and taken the gate low.



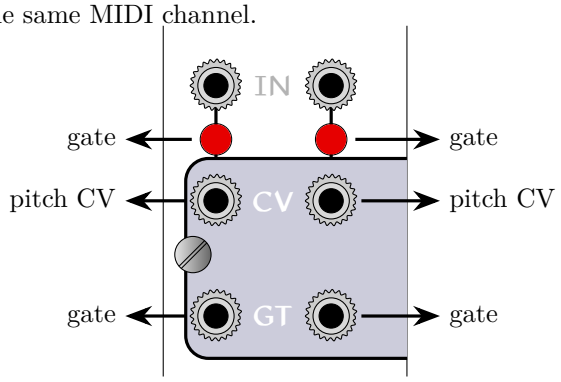
Channel 1 implements monophonic note stealing. If a new note starts while an old one is in progress, then the old note immediately ends and is replaced by the new one. In such a case the gate output goes low for approximately 1ms to signal the start of the new note.

A stolen note ends when it is stolen. For instance, if you play Note On C, Note On G, Note Off G, then the gate goes low and the oscillator continues playing G. The C does not come back when the G ends, even if you still have not yet played Note Off C.

Although in this mode the module does not *use* its tempo clock, it can *accept* clock signals on the input jacks. The left input jack accepts a standard 24 PPQN MIDI clock. The right input jack will function as a 1 PPQN or tap tempo input if there is nothing received on the left input, or as a synchronization or reset input (marking the start of a quarter note) when there is also a 24 PPQN clock being received on the left.

Channel 2: duophonic CV/gate

This mode is intended for controlling a synthesizer with two more or less identical voices. Each voice has a pitch CV and a gate CV, which respond to Note On and Off messages for up to two simultaneous notes in the same MIDI channel.



Channel 2 implements note stealing according to these rules:

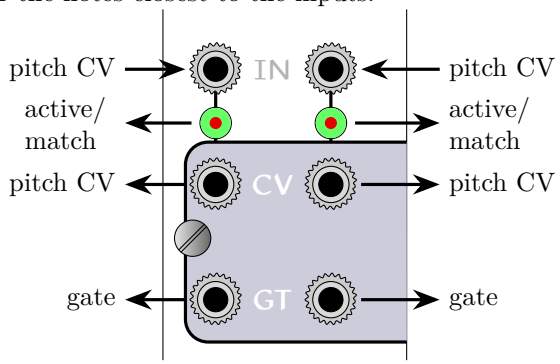
- The first note goes to the left.
- If exactly one side is free, a new note goes there.
- Otherwise (both sides free, or neither), a new note goes to the side that least recently had a new note.

As with Channel 1, the gate CV drops for about 1ms when a new note replaces an old one, so as to give some chance of retriggering whatever module is using that gate signal. Also as in Channel 1, stolen notes end when they are stolen.

Signals on the input jacks are ignored in this mode, and the LED on each side glows red when there is a note active on the side in question. Pitch bend in this channel affects both sides.

Channels 3 and 4: quantize to MIDI

When it receives MIDI notes in either of these two channels, the Gracious Host acts as a dual quantizer. The input voltage on each side is compared to whichever MIDI notes are currently held, and the output pitch control voltages are set to the voltages for the notes closest to the inputs.



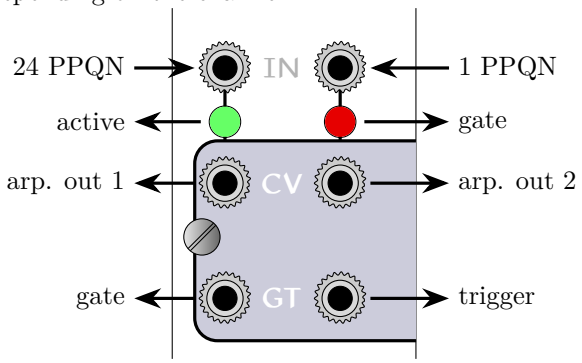
If, once in this mode, there happen to be no MIDI notes held, then all four output voltages go to zero and the LEDs go dark. Otherwise, the LED on each side is red when the input happens to hit a quantized output value exactly (that is, within the same semitone), and green otherwise. Each gate output is normally high when any notes are held, but it drops low for approximately 1ms when the associated pitch output goes to a new note, in order to allow for retriggering of a synthesizer voice.

The difference between the two channels is that with Channel 3, quantization is just to the received MIDI notes and no others. With Channel 4, each MIDI note counts as if you played it *in every octave*. So if you play MIDI notes 60 and 69 (Middle C and the A above it, equivalent to output voltages 2.00V and 2.75V) in Channel 3, then an input voltage of 0.50V (equivalent to MIDI note 42) will quantize to 2.00V (MIDI note 60); but with the same notes played in Channel 4, an input voltage of 0.50V will quantize to 0.75V (MIDI note 45) because playing any C and A activates every C and A as a possible quantization value.

Pitch bend, if nonzero, is applied to both outputs equally after quantization; it does not affect the input quantization boundaries. Some future version of the firmware may attempt to do something more intelligent with pitch bend, such as using it to support microtonal quantization.

Channels 5–7: arpeggiator modes

MIDI notes on these three channels will be arpeggiated to the CV and gate outputs, in different styles depending on the channel.



These channels use the global MIDI clock. MIDI Timing Clock messages, or pulses received on the left input jack, define the tempo at a rate of 24 PPQN. MIDI Start messages reset to the start of the quarter note. Pulses received on the right input jack, and the tap tempo key on a typing keyboard, reset to the start of the quarter note and also define the tempo when there is no 24 PPQN clock.

The left digital output is the gate; it goes high (9V nominal) and the right LED glows red, for the first 7/8 of each quarter note time. The right digital output produces a 960μs trigger at the start of each quarter note time; this can also function as a 1 PPQN clock for other modules. The left LED glows green when there are any held notes, that is, when the arpeggiator is active. As for the CV outputs, they cycle through the held MIDI notes in a way that depends on the channel.

Channel 5 arpeggiates up and down. The left output cycles through the notes upward (in order of increasing pitch) while the right output cycles downward.

Channel 6 arpeggiates the held notes in the order they were entered, with the right output one step

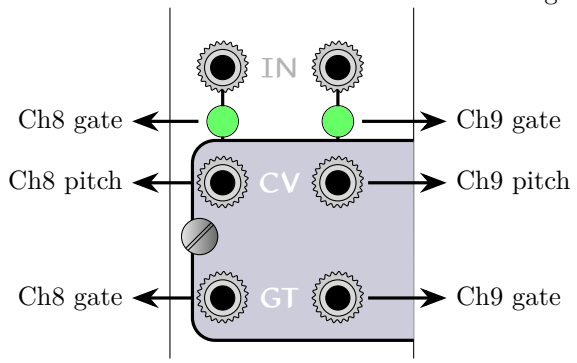
ahead of the left.

Channel 7 arpeggiates in random order. In more detail, the right output selects one of the held notes uniformly at random on each beat. That means *every* held note is available; the right output can repeat notes. The left output selects a note at random, but if there are at least two held notes then it will avoid choosing the same note as the right output, and if there are at least three held notes then it will also avoid its own previous value; other than those exceptions, it selects uniformly. So the left output will not repeat notes, given sufficient held notes to choose from. The random numbers for Channel 7 come from a hashed entropy pool continuously reseeded by I/O timings, and they should be random enough for rock'n'roll even if not cryptographically certifiable.

All three arpeggiation modes start their new notes *on the beat*. If you start playing notes into these channels when the tempo is slow, there may be a noticeable delay before the output starts playing; but this design feature is intended to allow playing more precisely *with* the beat in typical modular performance styles. Note that the tempo clock needs to be running in order for the arpeggiator to be useful. To use these channels you must provide the module with MIDI clock messages, a clock on the input jacks, or tap tempo from a typing keyboard.

Channels 8 and 9: dual mono

These two channels work together to allow independent control of two GV/gate synthesizer voices. Unlike Channel 2, which automatically assigns incoming notes to either channel, you can use this pair of channels to specify on which side each note will play. Channel 8 controls the left and Channel 9 the right.



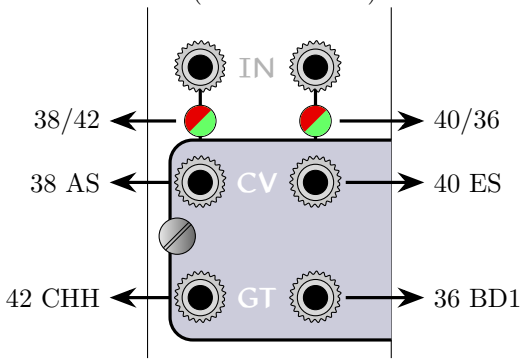
Much like Channel 1, these channels do monophonic note stealing. The LED for each channel glows green when a note is playing. The input jacks are ignored. Channels 8 and 9 have independent pitch bend.

Channels 10 and 11: drum notes

These channels map note numbers into pulses on the four output jacks. Channel 10 produces a trigger of approximately 1ms at the start of each note, whereas Channel 11 produces a gate pulse, high as long as the note lasts.

Note numbers translate to output jacks according to a somewhat complicated mapping that is designed to work without needing special configuration on many common MIDI controllers. As a simple starting point, this assignment of General MIDI drum notes to output jacks is one that works:

- MIDI note 36 (Bass Drum 1): lower right
- MIDI note 38 (Acoustic Snare): upper left
- MIDI note 40 (Electric Snare): upper right
- MIDI note 42 (Closed Hi Hat): lower left



Signals on the input jacks are ignored. Each LED glows if any notes on the corresponding side are active, red if the upper note (analog CV output) is active and green if the lower but not the upper note is active.

The upper analog outputs in drum mode go to their maximum voltage when high, which is 5.5V nominal (uncalibrated). The lower digital outputs, as usual, go to about 9V. The trigger pulse width is nominally 960μs.

Now, some more detail on how note mapping works. Every MIDI note maps to one of the four output jacks, so you can make up a mapping by choosing any four notes that happen to map to different jacks. If you have a controller like a keyboard with many notes on it, you can probably find a usable mapping quickly just through trial and error, but the scheme is designed to guarantee that:

- any four consecutive MIDI notes starting with a multiple of four, such as $\{0, 1, 2, 3\}$ or $\{60, 61, 62, 63\}$, will map to distinct jacks; and
- any four MIDI notes spaced two apart, such as

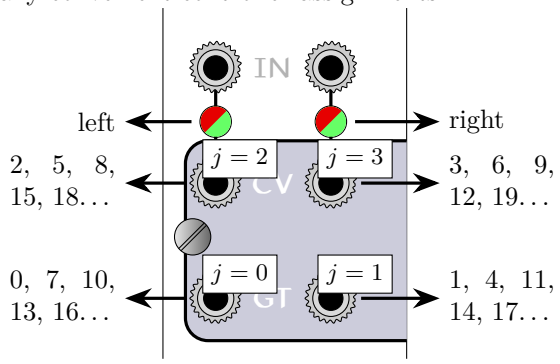
$\{0, 2, 4, 6\}$ or $\{65, 67, 69, 71\}$, will map to distinct jacks.

That means pad controllers which typically assign the pads to consecutive notes will often have four conveniently-arranged pads which control the four output jacks. It also means that the notes F, G, A, B (which are two semitones apart) in any octave on a piano-style keyboard, or any four horizontally adjacent keys on a Wicki-Hayden isomorphic keyboard (as discussed in the typing-keyboard chapter of this manual), will work.

In even more detail: each note number N maps to a jack number $j = (\lfloor N/4 \rfloor + N) \bmod 4$. In words, that formula says to start with the note number N, divide it by four and throw away the remainder, then add the original N, divide the result by four again, and this time throw away the quotient and look at only the remainder, which we'll call j. Then:

- if $j = 0$, the note activates the lower left jack;
- if $j = 1$, it activates the lower right jack;
- if $j = 2$, the upper left jack; and
- if $j = 3$, the upper right.

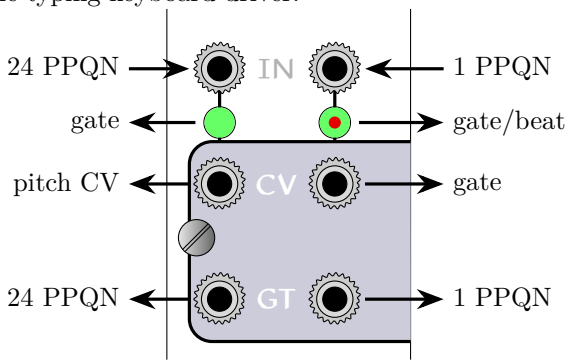
That splits the range of MIDI note numbers into four sets with the desired properties of being hit by many convenient controller assignments.



Channel 12: mono with clock out

This monophonic mode provides CV/gate to control a synthesizer voice, with output of the MIDI clock as Eurorack trigger signals. The gate output is sent through an analog output jack, meaning that its high level will be approximately 5.5V. The clock outputs are 960μs triggers with a high level of approximately 9V, at 24 PPQN and 1 PPQN. The 1 PPQN pulse is scheduled about 1ms before the first 24 PPQN pulse of the quarter note, so that it can be used as a "reset"; these pulses are meant to express the same semantics as the MIDI Timing Clock and MIDI Start messages.

The source for timing can be the input jacks, MIDI timing messages, or the tap tempo feature of the typing keyboard driver.



This channel implements monophonic note stealing, the same as Channel 1. Both LEDs glow green when a note is playing, but the right LED also flashes red on the beat, overriding the green.

Channels 13–16: reserved

The last four channels are not currently implemented. They are available for use by future versions of the official firmware, or possibly by user-defined firmware.

Typing keyboard interface

The Gracious Host supports two kinds of USB keyboards: musical keyboards with a USB MIDI interface as described in the previous chapter, and the other kind of keyboard that is commonly attached to a PC and used for typing. The driver for typing keyboards, described in this chapter, allows them to be used for playing synthesizers – with some limitations, but at a much lower cost than most full-featured MIDI keyboards.

General comments on USB keyboards

The typing keyboard driver supports what the USB standards call the "boot keyboard" protocol. In technical USB terms that means it will support a USB device with an interface descriptor of class 3, subclass 1, protocol 1. This is the standard USB keyboard protocol used by the typing keyboards people commonly plug into personal computers. It is called "boot keyboard" because the authors of the relevant standard apparently imagined that the simple protocol would only be used by the BIOS configuration menus of PCs, and most operating systems would instead use the ridiculously complicated full-generality "HID report protocol" instead; but in fact, most real-world implementations just use the boot keyboard protocol.

Wiring switches together to form a keyboard becomes more complicated if it is desired to correctly detect when the user presses more than one key at the same time. In musical keyboards this issue is called "polyphony"; with typing keyboards, the same thing is called "rollover." The USB boot keyboard protocol is only capable of reporting to the host at most six simultaneous keypresses (of regular typing keys; modifier keys like shift are handled separately and do not count toward this limit) because it sends six bytes of key information per report packet with each key consuming one byte. So a USB keyboard plugged into the Gracious Host can play *at most* six simultaneous MIDI notes in the ordinary way by pressing keys at once.

However, reaching the limit requires keyboard hardware actually capable of six-key rollover. Some keyboards, especially fancier ones marketed as "gaming" keyboards, are able to do it; but cheaper generic USB keyboards cannot really do full six-key rollover. The average USB typing keyboard that does not specifically advertise a rollover feature may give incorrect results (not detecting some keys, or spuriously detecting unpressed "ghost" keys) when multiple keys are pressed, usually depending in a complicated way on the specific key combination involved. The Gracious Host only knows what the keyboard tells it, and cannot easily compensate for such behaviour.

For the quantizer and arpeggiator features it's useful to play many notes at once; so to get around both keyboard rollover limits and the user's limited number of fingers, the Gracious Host typing-keyboard driver supports a *sustain* feature using the Caps Lock key to lock keys in a pressed state without needing to hold them down. See the section on that below.

USB typing keyboards have a limit on how fast they can be polled, and that limits how responsive to keystrokes anything controlled by a USB typing keyboard can be. The limit depends on the specific keyboard model (the keyboard decides how fast it will respond) but the default is usually 10ms intervals, for 100 polls per second. There can also be a further delay of a few milliseconds internal to the keyboard as the keyboard hardware scans the array of switches. Human beings are not capable of perceiving this amount of so-called latency in keystroke response but many believe they can, and users who hold that belief might prefer to use a USB MIDI keyboard instead of a typing keyboard. The USB MIDI protocol as implemented by the Gracious Host allows for sub-millisecond polling.

The general layout of a USB typing keyboard is reasonably standardized, but there are many small details that vary from one to another. The biggest distinction is between "US-style" keyboards, with a large L-shaped Enter key, and "ISO-style" keyboards with a narrower vertical Enter key and a few more small keys to accommodate the additional letters in non-English languages. The precise details of those keys, which letters are shown on which keycaps, the

locations of some punctuation marks like backslash, and so on, vary a lot. Keyboards do not report the details of their layout to a USB host, so the host has to guess, maintain a database of specific keyboard models, or be configured for it out-of-band.

The Gracious Host attempts to support all relevant keys that exist in most popular keyboard layouts used around the world, with reasonable guesses as to where they will be located; but be aware that it's possible your keyboard will not actually have all the keys shown in the diagrams in this manual, or that a few of them (especially around the edges of the main keyboard area) will not be located exactly where they are shown.

The keys labelled "GUI" (and not implemented to do anything, in the current firmware) are called that here because that's what they are called in the relevant USB standard. On most keyboards they are actually labelled with the logo of an operating system vendor, like Microsoft or Apple.

Piano-style keyboard layout

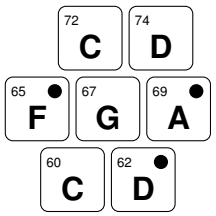
Figure 4 shows two layouts. The upper one is the default layout active when a keyboard is first plugged into the Gracious Host and the Num Lock LED is off. Note names and MIDI note numbers (assuming no octave shift) are as shown in the diagram.

This layout is meant to imitate a piano's keyboard layout, with a little over $2\frac{1}{2}$ octaves of coverage. Keys that fall into gaps of the piano layout (shown in grey) are assigned to B♯ and E♯ (enharmonic to C and F) to make a consistent pattern. The upper left of the main letter area (Q on a QWERTY layout) is Middle C, and the F key in a QWERTY layout, which often has a tactile guide bump on it, is effectively an F (shown as E♯, but the same MIDI number).

Wicki-Hayden isomorphic layout

Press Num Lock to switch layouts. With the Num Lock LED glowing, the Gracious Host implements a Wicki-Hayden isomorphic keyboard layout, as shown in the lower half of Figure 4. This layout is named after Kaspar Wicki and Brian Hayden, who independently invented and patented it in 1896 and 1986 respectively. It is similar to the layout commonly used for concertina keyboards.

The Wicki-Hayden layout treats the keys as an hexagonal grid. Horizontally adjacent keys are a major second (two semitones) apart, pitch going up from left to right across the keyboard. From any note the diagonally adjacent notes to the right represent the fifth of the note, in the higher or lower octave according to whether they are diagonally up and right or down and right. Similarly, the diagonal notes to the left are the fourth of the current note. And going directly up or down two rows corresponds to going up or down by a whole octave.



The defining feature of an *isomorphic*[*] music keyboard is that the harmonic relationships are the same everywhere on the keyboard. If you memorize the shape for a given chord such as D minor, shown by the dots above, you can play any chord of the same quality, such as C minor, by shifting the same shape somewhere else on the keyboard. These chords would have to be learned separately on a piano, where D minor is played entirely on white keys and C minor involves a black key. Melodies can be transposed on an isomorphic keyboard just by moving to a different physical location without changing the fingering.

With no octave shift, the Gracious Host's Wicki-Hayden layout puts the F and G above Middle C on the F and G keys of a standard QWERTY layout.

Sustain

The Caps Lock key activates the *sustain* feature. Modes like quantizer and arpeggiator benefit from being able to play many MIDI notes at once; but both because people usually have at most ten fingers, and because USB typing keyboards are limited in how many simultaneous keypresses they can handle, actually pressing many note keys at once may be a problem. The basic sustain function is that when Caps Lock is pressed, the associated LED goes on, and then all note keys pressed at that time or while Caps Lock remains pressed, are locked. These notes remain in effect even if the note keys are released. When Caps Lock is pressed a second time and the LED goes off, all the locked notes are considered released at once.

The usual performace practice would be to either hold a chord and tap Caps Lock to lock it; or to press and hold Caps Lock, and while holding it build up the chord one or a few notes at a time before releasing Caps Lock.

[*] This word means "same shape."

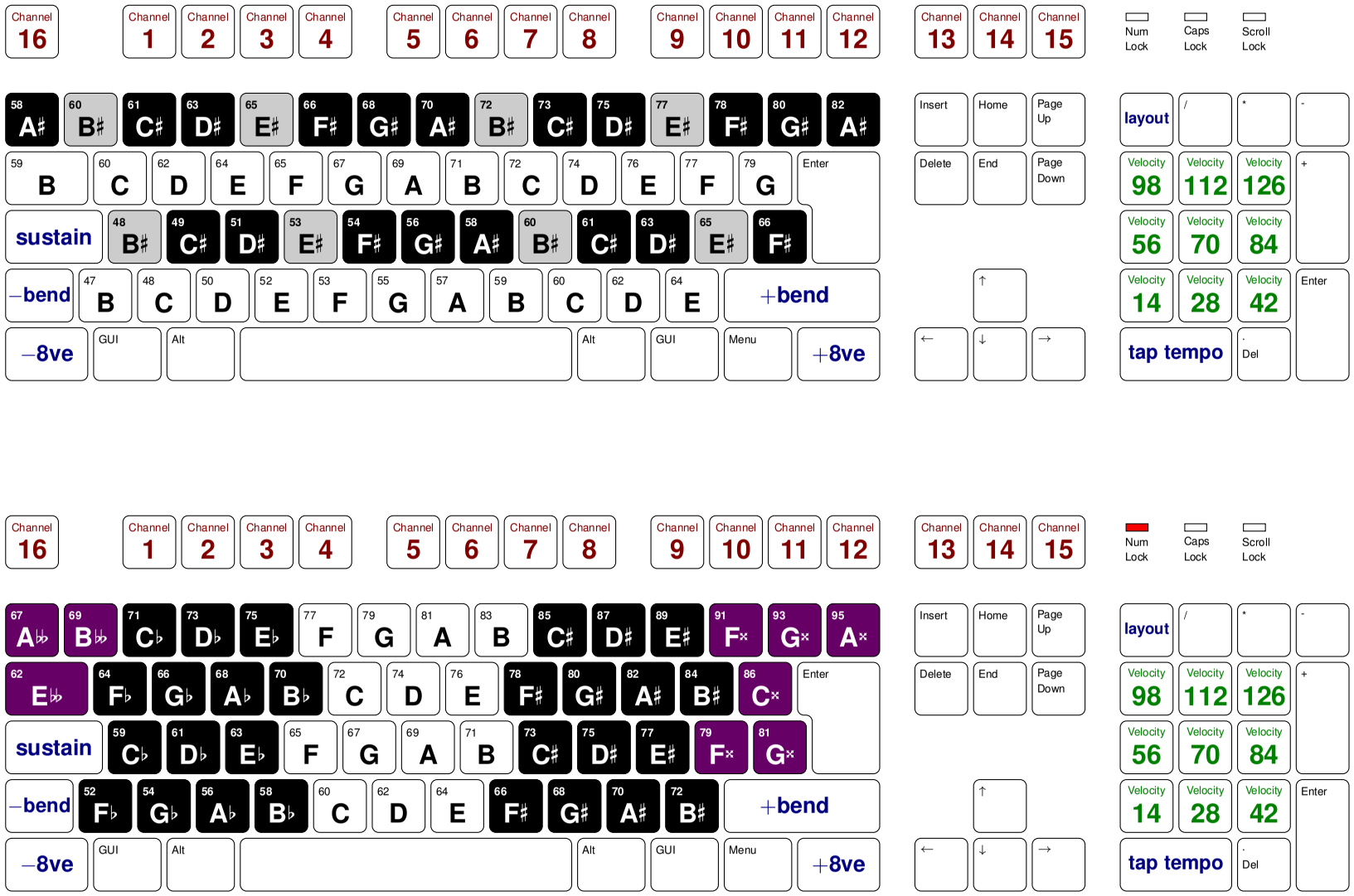


Figure 4: Keyboard layouts.

In more detail: Notes become locked if they are playing at the moment the Caps Lock key is first pressed to activate sustain. They also become locked if they are newly pressed *during* that first press of Caps Lock, so the sequence "press Caps Lock and hold; press note key; release Caps Lock" results in the note being locked, regardless of exactly when the note key is released. Once locked, notes remain locked, and no additional Note On events will be sent, until the second Caps Lock keypress, which will turn off the LED and send Note Off events all at once for all the locked notes except those that may actually be pressed at the moment of the second Caps Lock keypress (which instead will be released when the note keys are actually released). After the first time Caps Lock has been released, with the LED on and some notes held, other notes can be played normally. Replaying a note already held will have no additional effect – no second Note On will be sent – except that a note whose key is actually pressed when releasing sustain will not get a Note Off at the release of sustain, but only when its key is also released.

The sustain feature remembers the current channel when it is activated. If you change the channel while sustain is active, notes played in the new channel are completely separate. New Note On messages may be sent in the new channel, and the eventual Note Offs when sustain is released will be sent in the original channel. Also, sustain is associated with *note numbers*, not with *physical keyboard keys*. By changing the octave shift or the piano/isomorphic layout selection at different points in the use of the sustain feature, it is quite possible for the same note key to end up causing more than one note to be locked.

Octave shift

The two Ctrl keys on the USB keyboard control octave shift. Press the left Ctrl key (just press like a normal key; it is not necessary to hold it down, or press other keys along with it) to shift down one octave. The typing keys that send MIDI notes will send notes one octave lower than their default values. Press the right Ctrl to shift one octave up. The Scroll Lock LED will glow (as well as possibly blinking with the beat, see "tap tempo" below) whenever an octave shift in either direction is active.

Pressing the shift keys again, shifts further. Shifting up to five octaves down or four octaves up is allowed, those limits being chosen to allow covering the range of MIDI notes 1 to 127 in either keyboard layout, even on a US-style keyboard with relatively few keys. However, in practice it will seldom be useful to play notes outside the range 24...96, which corresponds to the range of the control voltage output DACs.

Pitch bend

The right and left Shift keys send MIDI pitch bend messages. The pitch will bend up or down while you hold either Shift key (cancelling out, if both) up to its default limit of two semitones, then will return once the Shift key is released. The speed of bend and return is fixed in the firmware; to achieve finer control of pitch bend you need a proper MIDI controller with this feature.

Channel selection

Each function key (F1, F2, and so on) corresponds to the MIDI channel with the same number. Press one of these keys to set the channel on which the typing keyboard will send subsequent MIDI events. See the previous chapter of this manual for descriptions of the different channels. You can, for instance, press F2 to switch to duophonic mode, or F10 to send drum triggers. Channel 1 is the default when the typing keyboard is first connected, before any function key has been pressed.

The remaining keys in the function-key row select the remaining channels: Print Screen, Scroll Lock, and Pause/Break (at the right of the row) correspond to Channels 13, 14, and 15 as if they were F13, F14, and F15; and Esc (usually at the left of the row, though it appears elsewhere on some keyboards) to Channel 16 as if it were F16. However, in the current firmware as of this writing, channels beyond 12 do not actually do anything.

Note that the MIDI subsystem determines its mode from *the channel of the most recent Note On message*. Just pressing a function key to change channels will not change the MIDI subsystem's mode; you must press a note key to send a Note On before the MIDI subsystem will change modes. The concept here is that the module implements a MIDI to CV interface. When you plug in a typing keyboard, the typing keyboard is just a funny-looking MIDI master keyboard plugged into the interface. The function keys are configuration commands to the keyboard regarding what channel it should send MIDI messages on in the future; they are not MIDI messages in themselves. The MIDI subsystem, to the extent possible, responds to MIDI messages from a typing keyboard just the same way it would respond to MIDI messages

from a music keyboard.

Velocity

The keypad numerals 1 through 9 set the velocity for MIDI Note On events. Press any of these to choose the velocity for any subsequent notes. The values are as shown in the keyboard layout diagram, equal to the keycap numeral value times 14, to provide equally spaced values throughout the MIDI range. Velocity is only relevant to Channel 1 (where it will be a control voltage appearing on the right analog output) with the default firmware, but some customized or future firmware might use velocity in other channels in some way.

Tap tempo

The keypad Insert (0) key functions as *tap tempo*. This allows the performer to establish a clock for the arpeggiator and sync modes without needing to patch in a clock control voltage signal. In some channel modes the resulting clock appears as a control voltage output and can be used to control other modules. Although there are some differences in the internal implementation, the tap tempo feature basically serves the same purpose as MIDI Timing Clock messages (24 of those per tap).

Press the tap tempo key at least twice, on the desired beat, to start the tempo clock. The Scroll Lock LED will blink on the beat (overlaid on its solid glow, if octave shift is active). If you enter three or more taps, in a reasonably consistent straight rhythm, the tempo will be determined by an exponential moving average of the most recent timings; that allows for a more precise fit than would be possible by using only the two most recent, given the limited precision of USB keyboard timing. A tap that is not close to the established timing, or that seems to have skipped at least one beat, will be treated as the start of a new tap tempo sequence, stopping the old clock.

Maintenance codes

It is possible to activate special features, mostly intended for firmware testing, by entering a four-digit decimal number through the typing keyboard driver. The only maintenance code that most module owners will find really useful is 5833, to run the calibration procedure. But for completeness, here is a list of all the codes supported by production firmware.

5833 Run the calibration procedure.

1240 Simulate USB hub insertion (blinks "H" in Morse code).

3627 Simulate insertion of an unsupported USB device (blinks "D" in Morse code).

4935 Perform the "success" display (normally done as the result of a completed calibration).

6697 Perform the "failure" display (normally the result of an aborted calibration).

8189 Throw a driver exception (flashes lights and waits for USB disconnect).

8605 Simulate a power-on reset.

Special firmware assembled with test routines included will also accept a few other codes to run the test routines, but those will have no effect on standard production firmware. See the *Gracious Host Programmer's Manual* for details on compiling test firmware, and how to create your own maintenance codes.

To enter a maintenance code:

- Attach a typing keyboard to the module.
- Press and hold one (either) of the Ctrl keys and one of the Alt keys.
- While holding Ctrl and Alt, type out the four digits of the maintenance code, on the numeric keypad.

Mouse interface

An ordinary USB mouse plugged into the Gracious Host makes a simple CV/gate controller. The connections and basic functions for this mode are as shown in Figure 5.

The standardization of USB mice is much like that of typing keyboards: the relevant standard includes a very complicated protocol and a simple one, and most implementors only use the simple one. In technical terms, the Gracious Host supports the "boot mouse" protocol, for USB devices that expose an interface descriptor of class 3, subclass 1, protocol 2. Nearly all commonly-available USB mice can operate under this protocol.

The basic function with a mouse attached is that the left and right mouse buttons send gates (to the Gracious Host's digital outputs) and the X and Y coordinates of mouse motion control the analog CV outputs. The middle button (or wheel, when clicked) cycles between the quantization modes below; and the colours of the LEDs (which light up on button presses) indicate the current mode.

- Unquantized (both LEDs green; default on startup) – CV outputs directly reflect the current X and Y coordinates.
- Smart quantize (both LEDs red) – CV outputs are quantized to a diatonic scale that attempts to automatically follow the notes you play. Play the fourth of the (major) scale three times without playing the seventh and it will shift to the subdominant key; play the seventh three times without playing the fourth, and it will shift to the dominant. In practice, this allows for free improvisation with the somewhat clumsy mouse control, while keeping everything more or less sounding like it is in tune.
- Quantize to fixed scale (left LED red, right green) – CV outputs are quantized to a fixed diatonic scale (C major or A minor, if 0V output is considered to be C).
- Quantize to semitones (left LED green, right red) – CV outputs are quantized to V/oct semitones, that is, round multiples of 1/12 of a volt.

In the quantization modes, the CVs are quantized only while the button is held and the gate is high; otherwise they are unquantized. That way, it is convenient to build a patch where one voltage is quantized pitch and the other is something like filter cutoff not meant to be quantized; to control the patch as intended, just don't press the button on the unquantized side.

The input jacks are not used in mouse mode, and (because the USB boot mouse protocol does not specify this) the mouse wheel's scrolling function, other than clicking it, has no consistent interpretation.

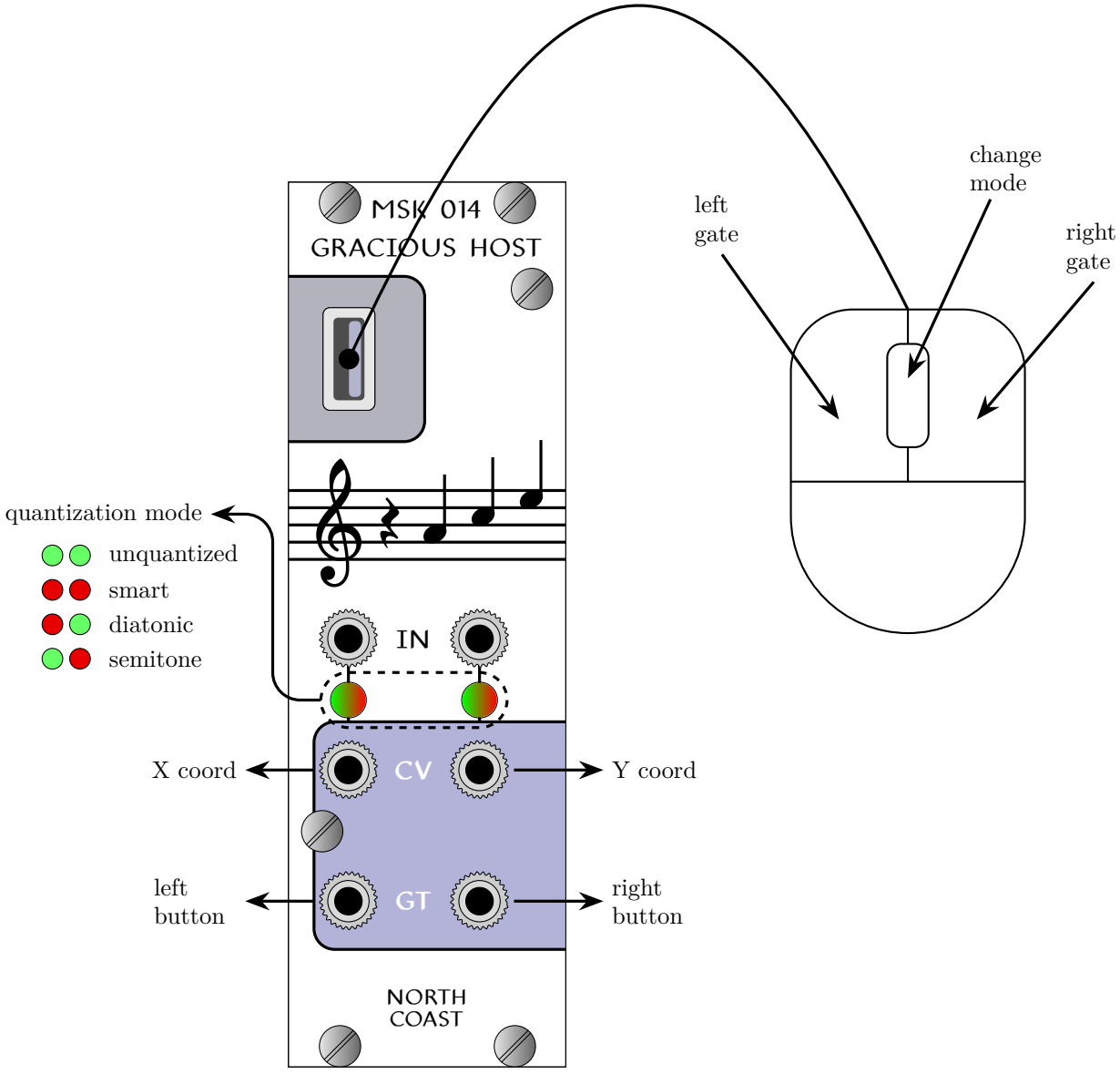


Figure 5: Mouse interface functions.

Safety and other warnings

Ask an adult to help you.

North Coast Synthesis Ltd. does not offer warranties or technical support on anything we did not build and sell. That applies both to modules built by you or others from the kits we sell, and to fully-assembled modules that might be built by others using our plans. Especially note that because we publish detailed plans and we permit third parties to build and sell modules using our plans subject to the relevant license terms, it is reasonable to expect that there will be modules on the new and used markets closely resembling ours but not built and sold by us. We may be able to help in authenticating a module of unknown provenance; contact us if you have questions of this nature.

For new modules purchased through a reseller, warranty and technical support issues should be taken to the reseller *first*. Resellers buy modules from North Coast at a significant discount, allowing them to resell the modules at a profit, and part of the way they earn that is by taking responsibility for supporting their own customers.

We also sell our products to hobbyists who enjoy tinkering with and customizing electronic equipment. Modules like ours, even if originally built by us, may be quite likely to contain third-party "mods," added or deleted features, or otherwise differ from the standard specifications of our assembled modules when new. Be aware of this possibility when you buy a used module.

The Gracious Host contains digital circuits that are sensitive to electrostatic discharge (ESD). The parts before assembly are more sensitive than the completed module, but even the completed module can be damaged by static. Take anti-static precautions when assembling, handling, or shipping it.

The configuration jumpers on the back of the Gracious Host must be set correctly for it to operate. Two Gracious Hosts plugged into the same bus, both with the jumpers in the default configuration, will conflict with each other.

This module requires +5V power.

Soldering irons are very hot.

Solder splashes and cut-off bits of component leads can fly a greater distance and are harder to clean up than you might expect. Spread out some newspapers or similar to catch them, and wear eye protection.

Lead solder is toxic, as are some fluxes used with lead-free solder. Do not eat, drink, smoke, pick your nose, or engage in sexual activity while using solder, and wash your hands when you are done using it.

Solder flux fumes are toxic, *especially* from lead-free solder because of its higher working temperature. Use appropriate ventilation.

Some lead-free solder alloys produce joints that look "cold" (i.e. defective) even when they are correctly made. This effect can be especially distressing to those of us who learned soldering with lead solder and then switched to lead-free. Learn the behaviour of whatever alloy you are using, and then trust your skills.

Water-soluble solder flux must be washed off promptly (within less than an hour of application) because if left in place it will corrode the metal. Solder with water-soluble flux should not be used with stranded wire because it is nearly impossible to remove from between the strands.

Voltage and current levels in some synthesizer circuits may be dangerous.

Do not attempt to make solder flow through the board and form fillets on both sides of every joint. Some soldering tutorials claim that that is desirable or even mandatory, it does look nicer, and it may happen naturally when the conditions are good. But a well-made solder joint that just covers the pad and makes good contact to the lead on one side of the board, is good enough.

Building your own electronic equipment is seldom cheaper than buying equivalent commercial products, due to commercial economies of scale from which you as small-scale home builder cannot benefit. If you think getting into DIY construction is a way to save money, you will probably be disappointed.

Bill of materials

This table is not a substitute for the text instructions.

Qty	Ref	Value/Part No.	
4	C13, C14, C19, C20	100pF	radial ceramic, 0.2″ lead spacing
10	C1, C2, C8–C10, C12, C15–C18	0.1μF	axial ceramic
4	C3, C4, C6, C7	10μF	radial aluminum electrolytic, 0.08″ lead spacing
1	C11	10μF	super high-value ceramic
1	C5	150μF	radial aluminum electrolytic
4	D1–D4	1N5818	or SB130; Schottky rectifier
2	D5, D6	MCL056PURGW	bi-colour LED, Multicomp
1	F1	200mA	polymer “fuse”
2	H5, H6		nut for M3 machine screw
2	H7, H8	M3x11	M3 male-female standoff, 11mm body length
2	H9, H10	M3x13	M3 male-female standoff, 13mm body length
4	H11–H14		nylon washer for M3 machine screw
6	H15–H20	M3x6	M3 machine screw, 6mm body length
6	J1–J6	1502 03	switched mono 3.5mm panel jack, Lumberg
2	J7, J8		female single-row socket, 10 pins at 0.1″
1	J9		male header for jumpers, 2 × 5 pins at 0.1″
1	J11	USBA	vertical USB-A connector
1	P1		male Eurorack power header, 2 × 8 pins at 0.1″, right angle
2	P2, P3		male single-row header, 10 pins at 0.1″
2	R15, R17	270Ω	
4	R25, R26, R29, R30	1kΩ	
2	R19, R20	2kΩ	
1	R14	2.4kΩ	
2	R9, R11	8.2kΩ	
6	R13, R21, R22, R31–R33	10kΩ	
2	R10, R12	12kΩ	
4	R23, R24, R27, R28	18kΩ	
2	R7, R8	27kΩ	
4	R1, R2, R16, R18	100kΩ	
2	R5, R6	330kΩ	
2	R3, R4	390kΩ	
1	U1	PIC24FJ64GB002	Microchip 16-bit microcontroller
1	U2	MCP4822	dual 12-bit DAC
1	U3	23LC1024	128Kbyte SRAM
2	U4, U5	LM224	quad bipolar op amp
1	U6	MCP1700-3302	3.3V LDO voltage regulator

Qty	Ref	Value/Part No.	
2	U4, U5		14-pin DIP socket
1	Y1	4.00 MHz	miniature DIP 3.3V clock oscillator

Fixed resistors should be 1% metal film throughout. RoHS-certified zinc-plated steel hardware is recommended, not stainless steel because of galvanic-corrosion incompatibility with aluminum parts.

Also needed: solder and related supplies, two PCBs, three configuration jumpers, front panel, 16-pin Eurorack power cable, etc.

Optional parts that may be added for development (not included in kits):

Qty	Ref	Value/Part No.	
1	P4		male single-row header, 6 pins at 0.1″
1	U7	7805	+5V regulator in TO-220 package

In some builds and kits the LM224 op amp chips may actually be LM224A.

Building Board 2

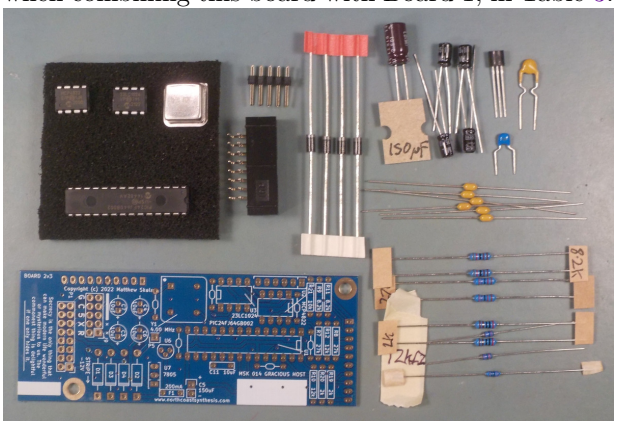
The recommended order for building this module is to assemble Board 2, the one further from the front panel, first. That will make it easier to get all the physical positioning right for the components that bridge between the boards or pass through the panel.

Note that although I'm describing a separate step for each component value, and that's how I built my prototype so as to have plenty of photo opportunities, if you are reasonably confident about your skills you may find it easier to populate all or most of the board (i.e. put the components in place) and then solder them in a single step. Except where noted, the order in which you add components does not matter much.

The digital ICs on this board are sensitive to static electricity. Although they are not *extremely* delicate, it would be appropriate to take precautions like working on an anti-static surface and wearing a grounding strap.

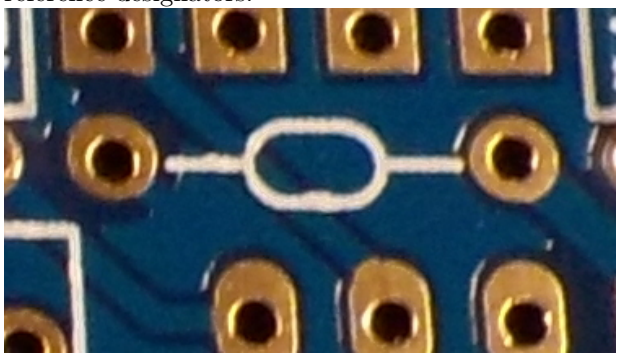
Preliminaries

Count out the right number of everything according to the bill of materials. There is an abbreviated BOM for Board 2, excluding a few items that will be added when combining this board with Board 1, in Table 3.

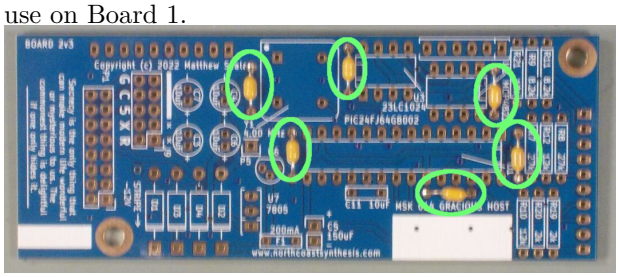


Decoupling capacitors

The six axial ceramic 0.1μF decoupling capacitors are shown on the board by a special symbol without their reference designators.



Install these six capacitors where the symbol appears. They are not polarized and may be installed in either orientation. These capacitors act as filters for the power supplies to the ICs. An MSK 014 kit should include ten of these capacitors, and only six are used on this board; save the remaining four for use on Board 1.



Fixed resistors

Resistors are never polarized. I like to install mine in a consistent direction for cosmetic reasons, but this is electrically unnecessary. In this module, the fixed resistors are metal film 1% type. They usually have blue bodies and four colour bands designating the value, plus a fifth band for the tolerance. The tolerance band is brown for 1%, but note that we may occasionally ship better-tolerance resistors in the kits than the specifications require, if we are able to source them at a good price. Accordingly, I mention only the four value band colours for this type of resistor; if you are using resistors with other codes, you are responsible for knowing them. Note that colour codes on metal film 1% resistors are often ambiguous (reading

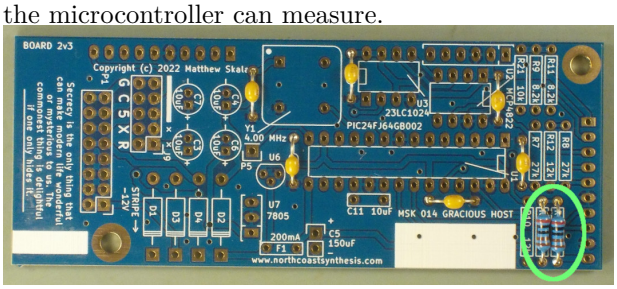
This table is not a substitute for the text instructions.

Qty	Ref	Value/Part No.	
6	C8–C10, C12, C15, C16	0.1μF	axial ceramic
4	C3, C4, C6, C7	10μF	radial aluminum electrolytic, 0.08″ lead spacing
1	C11	10μF	super high-value ceramic
1	C5	150μF	radial aluminum electrolytic
4	D1–D4	1N5818	or SB130; Schottky rectifier
1	F1	200mA	polymer “fuse”
1	J9		male header for jumpers, 2 × 5 pins at 0.1″
1	P1		male Eurorack power header, 2 × 8 pins at 0.1″, right angle
2	R19, R20	2kΩ	
2	R9, R11	8.2kΩ	
1	R21	10kΩ	
2	R10, R12	12kΩ	
2	R7, R8	27kΩ	
1	U1	PIC24FJ64GB002	Microchip 16-bit microcontroller
1	U2	MCP4822	dual 12-bit DAC
1	U3	23LC1024	128Kbyte SRAM
1	U6	MCP1700-3302	3.3V LDO voltage regulator
1	Y1	4.00 MHz	miniature DIP 3.3V clock oscillator

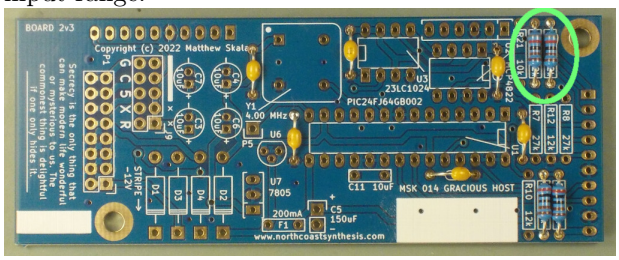
Table 3: Bill of Materials for assembling Board 2. Also needed is the PCB itself.

from one end or the other end may give two different values, both plausible) and some of the colours are hard to distinguish anyway. If in doubt, always measure with an ohmmeter before soldering the resistor in place.

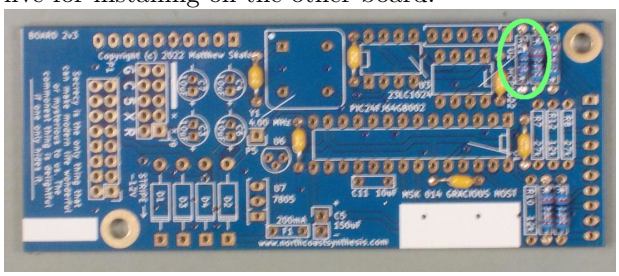
Install the two 2kΩ (red-black-black-brown) resistors R19 and R20. These resistors form a voltage divider that scales the USB power voltage into a range the microcontroller can measure.



Install the two 8.2kΩ (grey-red-black-brown) resistors R9 and R11. These form parts of the resistor networks that scale the analog input voltages (amplified on the other board) into the microcontroller's input range.

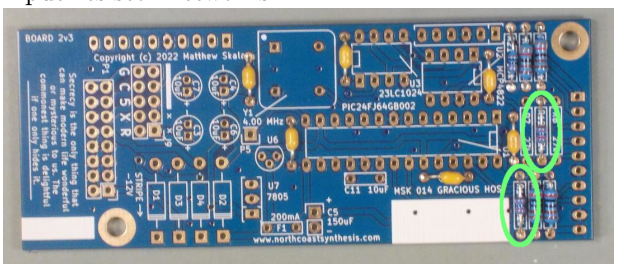


Install the single 10kΩ (brown-black-black-red) resistor R21. This is a pull-up resistor for the microcontroller's hardware reset input, which can be overridden via the in-circuit debugging port. There are six 10kΩ resistors in the module; reserve the other five for installing on the other board.

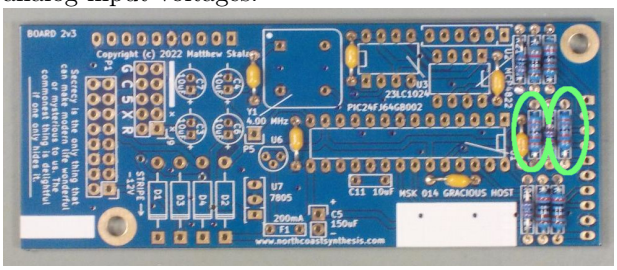


Install the two 12kΩ (brown-red-black-red) resis-

tors R10 and R12. These are also parts of the analog input resistor networks.



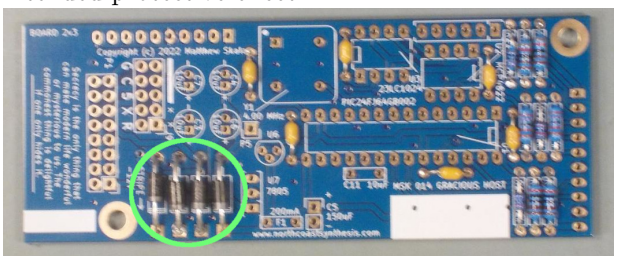
Install the two 27kΩ (red-violet-black-red) resistors R7 and R8. These, too, are used to scale the analog input voltages.



Semiconductors

Install the four 1N5818 or SBA130 Schottky rectifier diodes D1 through D4. These in general terms ensure that the different power voltages have the proper relationships: not connected backwards, and not a lower voltage connected while a higher voltage is disconnected; protecting the ICs and the onboard regulators.

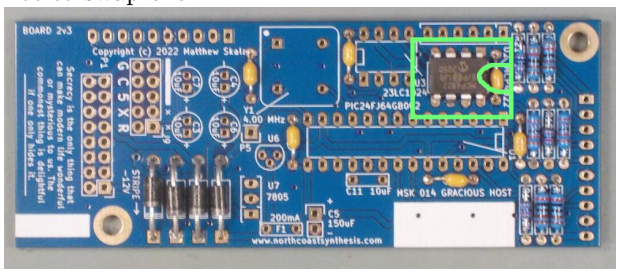
The diodes are polarized and it is important to install them in the right direction. Each diode is packaged inside a black or dark grey plastic slug with a white or light grey stripe at one end; that end is the *cathode*. The silkscreen markings on the board have a corresponding stripe and the diodes should be installed with their stripes matching the markings on the board. The solder pads for the cathodes are also square instead of round. Installing the diodes backwards means they will have the opposite of the intended protective effect.



The DIP ICs on this board are intended to be installed *without sockets*. These ICs send and receive signals at frequencies of up to 12 MHz (harmonics higher than that), and the microcontroller in particular is sensitive to the stray capacitance and inductance of its connections to other components. Mounting it without a socket reduces those stray reactances and improves the ability of the decoupling capacitors to keep everything stable. North Coast Synthesis kits for this module include some DIP sockets, but those are meant to be used for the op amp ICs on Board 1, not on this board.

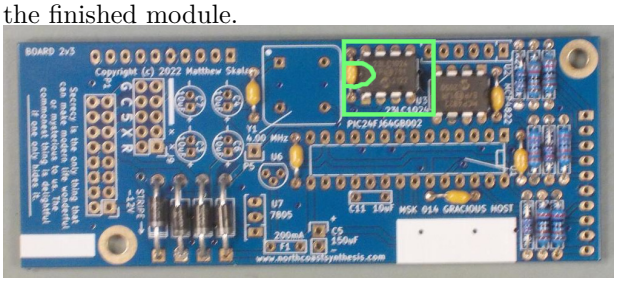
Because these ICs are installed without sockets, it is especially important to make sure that they are installed right way round. If you solder one in backwards, you will find it difficult to desolder without damage to correct the errors. *Double check the orientation of each IC before soldering.* Each IC has a notch at one end, or possibly a dimple in one corner, marking the location of Pin 1. On the PCB silkscreen, there is a marking indicating the location of the notch, and the pad for Pin 1 is rectangular instead of rounded.

Install the MCP4822 DAC chip, U2. This chip converts digital signals from the microcontroller to analog voltages for the analog output jacks. Make sure that its Pin 1 marking matches that on the circuit board, pointing up, toward what will be the top of the finished module. Also, make sure you are really installing the MCP4822. It is the same size and shape as the SRAM chip (next) and it is important not to swap them.



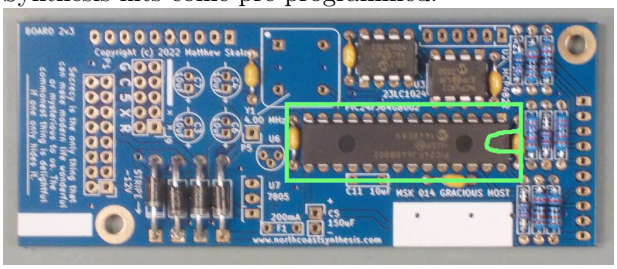
Install the 23LC1024 SRAM chip, U3. This chip provides additional memory space for the microcontroller, which is used to buffer firmware images during the firmware update process, as well as potentially for other purposes. Make sure its Pin 1 marking matches that on the circuit board, pointing *down* (opposite to the previous chip) toward what will be the bottom of

the finished module.



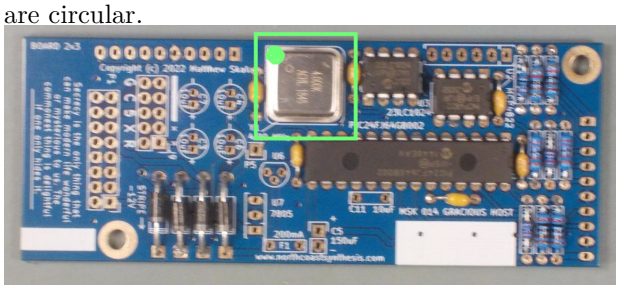
Install the PIC24FJ64GB002 microcontroller chip, U1. This chip is the heart and brain of the module, containing a small computer, associated peripherals, and the firmware that tells the computer what to do. It is in a long thin 28-pin "skinny DIP" package, and it may take some careful adjustment or bending of the legs to get them all nicely fitted into the holes on the board. Make sure the Pin 1 marking matches that on the circuit board, pointing up toward what will be the top of the finished module.

Note that the microcontroller chip needs to be *already programmed* before you install it; once installed, it can only be loaded with firmware through the USB port if it already has basically functional firmware on it, or through in-circuit debugging with the installation of an additional header and a PIC-specific programming tool. Chips shipped in North Coast Synthesis kits come pre-programmed.



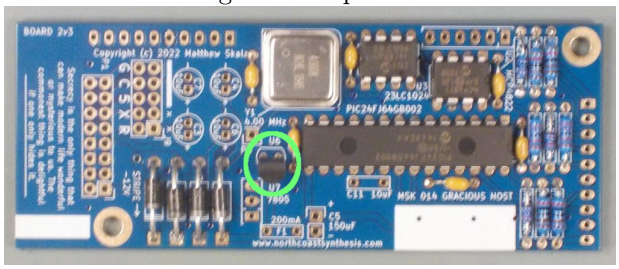
Install the 4.000 MHz clock oscillator module Y1. This is a hybrid device including an oscillator chip, a quartz crystal, and some support components, all inside a sealed metal can. It provides a stable and accurate reference frequency which will be multiplied and divided by the microcontroller chip to provide timing references for all parts of the Gracious Host.

The oscillator mounts in four holes arranged in a square pattern (basically the four corner holes of an 8-pin DIP footprint) and that makes orientation important. It can fit into the board in four different orientations, only one of which is correct. One corner of the oscillator will be marked, probably with a sharp point on the can where the other three corners are rounded, and a dot etched into the upper surface near the special corner. That corner represents Pin 1. The sharp corner and dot are shown on the PCB silkscreen and the oscillator should be mounted to match its markings with the silkscreen. The solder pad for Pin 1 is also square, whereas the other three are circular.



Install the MCP1700-3302 voltage regulator chip, U6. This is an LDO (low drop out) regulator that reduces the +5V Eurorack power supply to +3.3V for the digital chips. The PIC24 actually runs on +2.55V internally, and has a built-in regulator of its own to extract that from the +3.3V input.

The voltage regulator uses a TO-92 package, which is a small epoxy pill like a transistor package. The package has one flat side, and it should be installed with that flat side matching the flat side shown in the PCB silkscreen, facing the oscillator module. Bend the middle lead of the TO-92 backward to match the triangular hole pattern on the board.

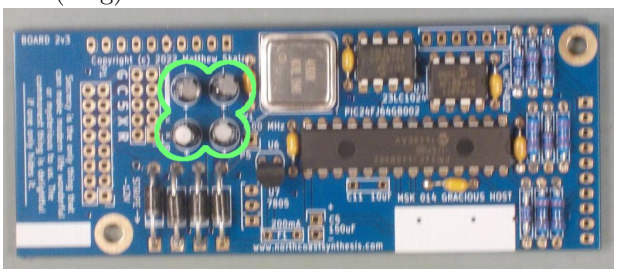


There are three pads on this board labelled "U7 7805," near the LDO regulator just installed, for installing an optional +5V regulator. See the comments in the general notes chapter of this manual for more information about installing this regulator. In most builds, it is not recommended.

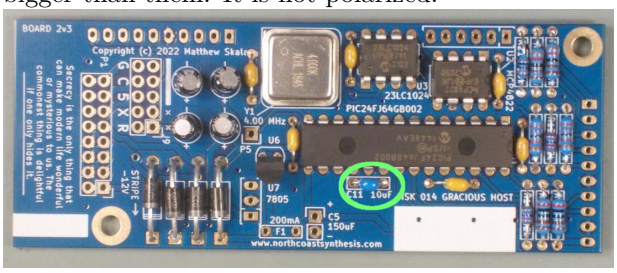
Capacitors and fuse

Install the four 10μF aluminum electrolytic capacitors C3, C4, C6, and C7. These are used for general filtering of the main power supplies: ±12V, +5V, and +3.3V, mostly for relatively low frequency noise. They are polarized components and they may explode if installed backwards. Each one will be marked on its casing with a stripe and minus signs to indicate the negative lead; the positive lead will probably also

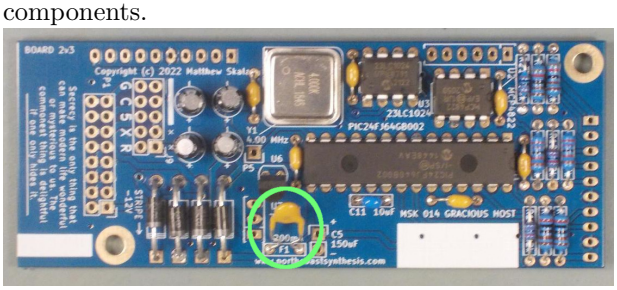
be longer. These clues should be matched with the markings on the PCB: plus and minus symbols in the silkscreen and a square solder pad for the positive (long) lead.



Install the 10μF ceramic capacitor C11. This is a special high-frequency filter for the PIC24's built-in core voltage regulator. Be sure not to confuse this capacitor with the other ceramic capacitors; it is about the same physical size as the 100pF capacitors used on the other board, but electrically it is very much bigger than them. It is not polarized.

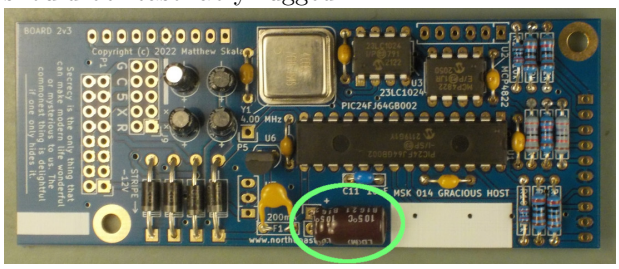


Install the 200 mA "polyfuse" F1. This device looks like a ceramic disc capacitor with kinks in the leads, but is actually a special kind of temperature-sensitive resistor. It serves to protect the USB power supply line from short circuits. It is unpolarized. Install it in the pads labelled F1. If you bend the leads to keep it from sticking out too far, then be careful not to leave it resting in direct contact with other components.



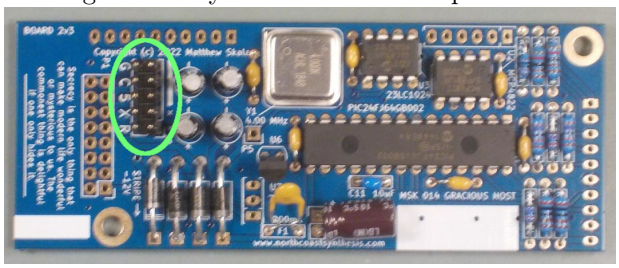
Install the 150μF aluminum electrolytic capacitor C5. This capacitor provides a substantial reservoir of charge for buffering transients as USB devices are plugged and unplugged. It looks like a physically larger version of the smaller aluminum electrolytics installed earlier, and like them, it is polarized and marked with a stripe on the body for negative and a longer lead for positive. The two pads on the board are labelled + and − to indicate which is which, and the + pad is square.

You will probably find it most convenient to lay the 150μF capacitor on its side on the board, with the leads bent down to go into the pads. You may need to secure it to the board with putty or tape to hold it in place while soldering, but once soldered it should be reasonably rugged.



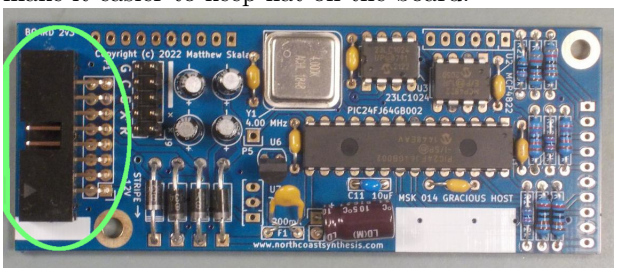
Header connectors

Install the 2×5-pin header connector J9. This connector is used with configuration jumpers to select whether the module should or should not connect to the Eurorack CV/gate bus, and the source of the module's +5V power. Try to install it snugly against the board, not tilted at an angle. Use tape or putty to hold it in place, solder one pin, then check that it is straight before you solder the other pins.



Install the 2×8-pin shrouded power connector P1. This is used for connecting the module to the Eurorack bus, for power and CV/gate output. As with the jumper block, it may be useful to solder just one pin and check the angle before soldering the rest, though the much larger body of this connector will probably

make it easier to keep flat on the board.



The board-to-board header connectors P2 and P3 will be installed on this board later, during the build of Board 1. There is also a footprint on this board for a header connector named P4, which is for in-circuit debugging and not installed in a standard build.

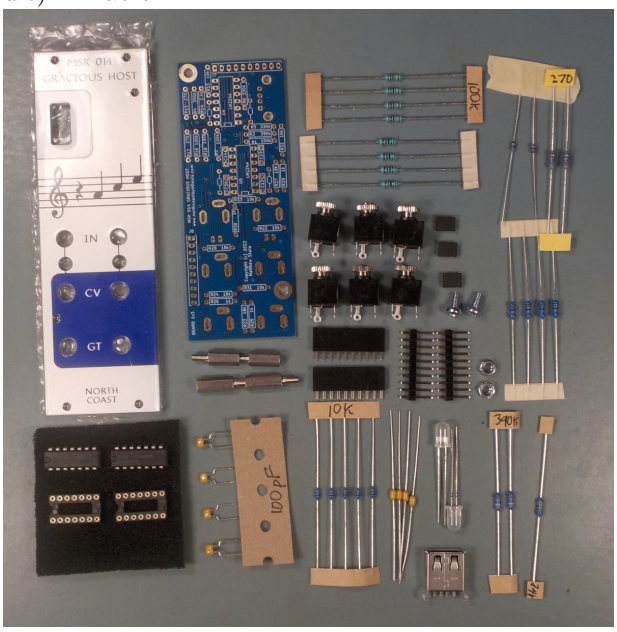
In between completed boards is a good time to take a break.

Building Board 1

Board 1 has components on both sides, and for best results, it is important to install them in the right order. Build Board 2 first, and see the general comments in the Board 2 chapter about how to approach the task.

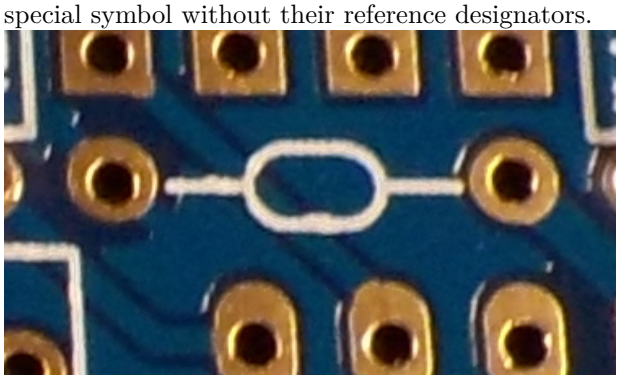
Preliminaries

Count out the right number of everything according to the bill of materials. There is an abbreviated BOM for the items needed in this chapter (including the connection to Board 2 and final assembly of the module) in Table 4.

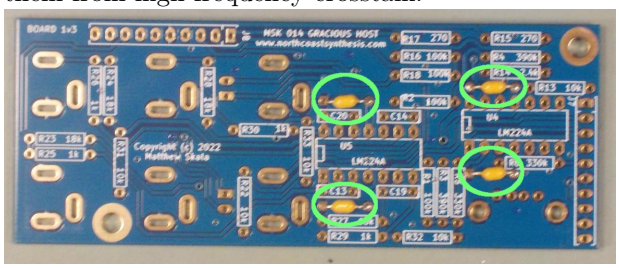


Decoupling capacitors

The four axial ceramic 0.1μF decoupling capacitors C1, C2, C17, and C18 are shown on the board by a special symbol without their reference designators.



Install these four capacitors where the symbol appears. They are not polarized and may be installed in either orientation. These capacitors act as filters for the power supplies to the op amp chips, protecting them from high-frequency crosstalk.



Fixed resistors

Resistors are never polarized. I like to install mine in a consistent direction for cosmetic reasons, but this is electrically unnecessary. In this module, the fixed resistors are metal film 1% type. They usually have blue bodies and four colour bands designating the value, plus a fifth band for the tolerance. The tolerance band is brown for 1%, but note that we may occasionally ship better-tolerance resistors in the kits than the specifications require, if we are able to source them at a good price. Accordingly, I mention only the four value band colours for this type of resistor; if you are using resistors with other codes, you are responsible for knowing them. Note that colour codes on metal film 1% resistors are often ambiguous (reading from one end or the other end may give two different values, both plausible) and some of the colours are

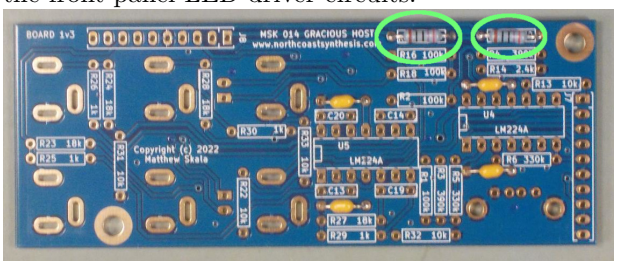
This table is not a substitute for the text instructions.

Qty	Ref	Value/Part No.	
4	C13, C14, C19, C20	100pF	radial ceramic, 0.2″ lead spacing
4	C1, C2, C17, C18	0.1μF	axial ceramic
2	D5, D6	MCL056PURGW	bi-colour LED, Multicomp
2	H5, H6		nut for M3 machine screw
2	H7, H8	M3x11	M3 male-female standoff, 11mm body length
2	H9, H10	M3x13	M3 male-female standoff, 13mm body length
2	H19, H20	M3x6	M3 machine screw, 6mm body length
6	J1–J6	1502 03	switched mono 3.5mm panel jack, Lumberg
2	J7, J8		female single-row socket, 10 pins at 0.1″
1	J11	USBA	vertical USB-A connector
2	P2, P3		male single-row header, 10 pins at 0.1″
2	R15, R17	270Ω	
4	R25, R26, R29, R30	1kΩ	
1	R14	2.4kΩ	
5	R13, R22, R31–R33	10kΩ	
4	R23, R24, R27, R28	18kΩ	
4	R1, R2, R16, R18	100kΩ	
2	R5, R6	330kΩ	
2	R3, R4	390kΩ	
2	U4, U5	LM224	quad bipolar op amp
2	U4, U5		14-pin DIP socket

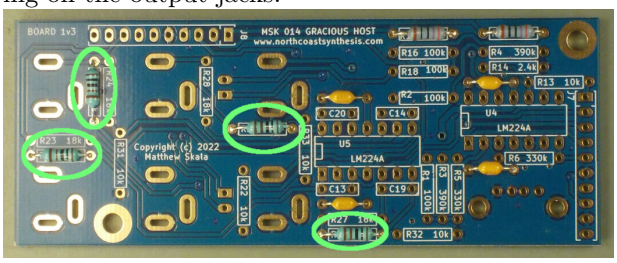
Table 4: Bill of Materials for Board 1. Also needed: the PCB itself, the aluminum front panel, the assembled Board 2, three configuration jumpers, and panel-to-rack mounting hardware.

hard to distinguish anyway. If in doubt, always measure with an ohmmeter before soldering the resistor in place.

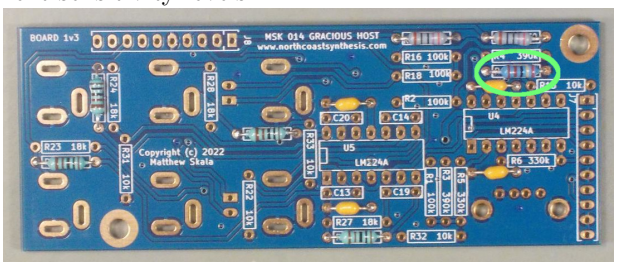
Install the two 270Ω (red-violet-black-black) resistors R15 and R17. These set the current level for the front-panel LED driver circuits.



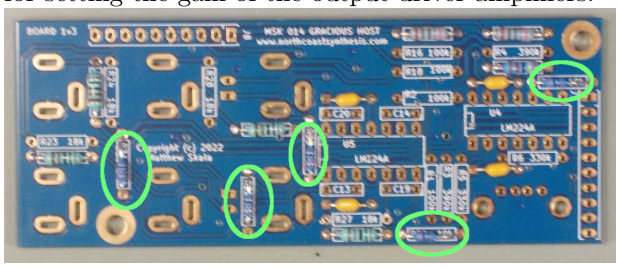
Install the four 1kΩ (brown-black-black-brown) resistors R25, R26, R29 and R30. These are current-limiting resistors to protect the output drivers, and other modules, in case of short circuits or bad patching on the output jacks.



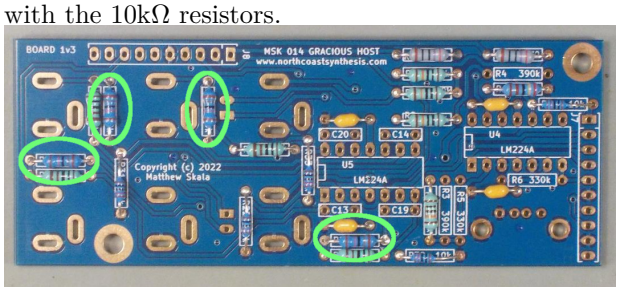
Install the single 2.4kΩ (red-yellow-black-brown) resistor R14. This is part of a voltage divider (with R13) that sets the relative balance of current levels between the red and green LED colours, to make them roughly equally bright given their different current sensitivity levels.



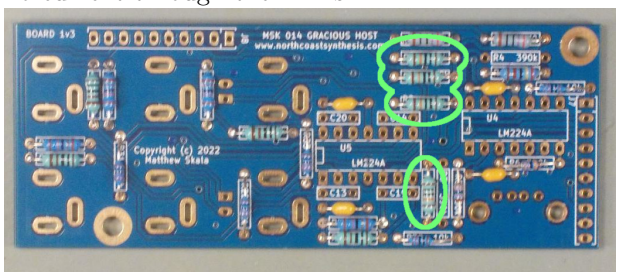
Install the five 10kΩ (brown-black-black-red) resistors R13, R22, and R31 to R33. R13 is part of the voltage divider with R14, and the other four are used for setting the gain of the output driver amplifiers.



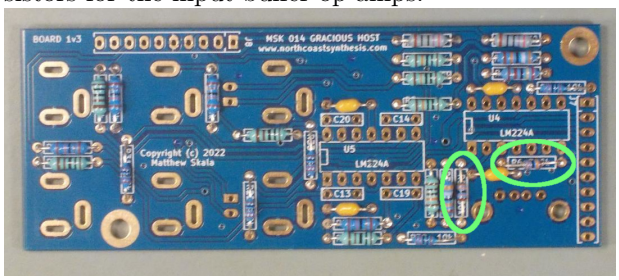
Install the four 18kΩ (brown-grey-black-red) resistors R23, R24, R27, and R28. These are feedback resistors for the output driver amplifiers, setting the voltage gain (to 2.8 non-inverting) in conjunction with the 10kΩ resistors.



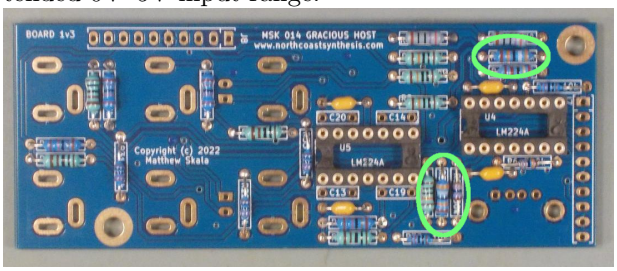
Install the four 100kΩ (brown-black-black-orange) resistors R1, R2, R16, and R18. The first two of these, R1 and R2, are input resistors that set the input impedance for the CV input jacks. The others, R16 and R18, set the default input voltage of the LED drivers when the microcontroller disconnects, so that an "LED off" command will correspond to little or no current through the LEDs.



Install the two 330kΩ (orange-orange-black-orange) resistors R5 and R6. These are feedback resistors for the input buffer op amps.



Install the two 390kΩ (orange-white-black-orange) resistors R3 and R4. These apply an offset to the input buffers, with the effect of shifting the op amp's clipping range to cover just beyond the intended 0V–5V input range.

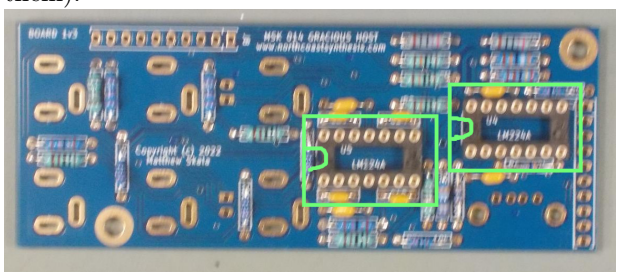


DIP sockets

Install the 14-pin DIP sockets for the LM224 quad operational amplifier (op amp) chips U4 and U5. The boards or chips may be labelled LM224A, with or without some other alphabetic suffix; the plain and -A versions have slightly different specifications but both will work in this circuit. Of the eight amplifier units on these two chips, two are used as LED drivers, and the rest are assigned one each to the front panel patching jacks: two input buffers and four output drivers.

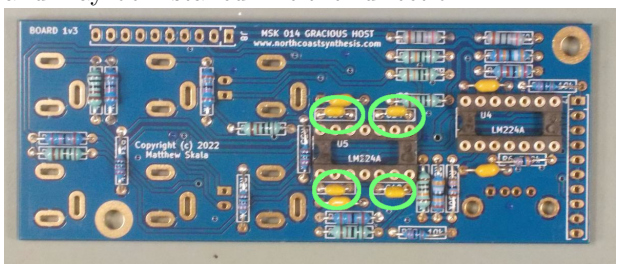
DIP sockets themselves do not care which direction you install them, but it is critically important that the chip installed in the socket should be installed in the right direction. To help with that, the socket will probably be marked with notches at one end (indicating the end where Pin 1 and Pin 14 are located) and you should install the socket so that the notched end matches the notch shown on the PCB silkscreen. The solder pad for Pin 1 is also distinguished by being rectangular instead of rounded.

Installing DIP sockets without having them tilted at a funny angle can be tricky. I recommend inserting the socket in the board, taping it in place on the component side with vinyl electrical tape or sticking it there with a small blob of putty at each end, then soldering one pin on one corner and checking that the socket is snug against the board before soldering the other pins. That way, if you accidentally solder the first pin with the socket tilted, it will be easier to correct (only one pin to desolder instead of all of them).



Compensation capacitors

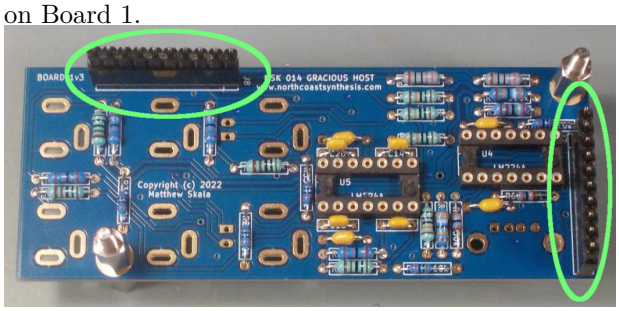
Install the 100pF radial ceramic capacitors C13, C14, C19, and C20. These are meant to ensure stability of the output drivers (one each). The capacitors will probably be marked "101," for 1 0 followed by one more 0, number of picofarads. They are not polarized and may be installed in either direction.



Board to board connectors

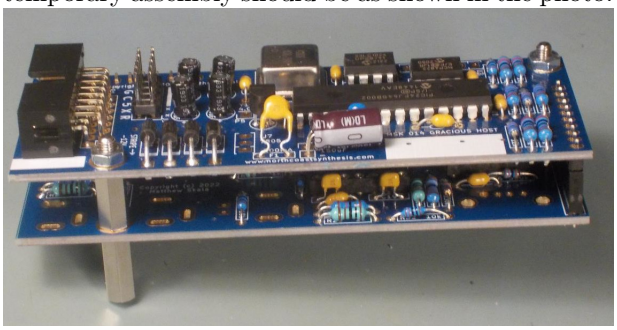
Fasten the two 13mm standoffs on the back of Board 1; that is the side opposite the components already installed. Attach them with the two 11mm standoffs. The male ends of the 13mm standoffs should pass through the mounting holes in the board and mate with the female ends of the 11mm standoffs on the front or component side of the board.

Mate the 10×1 header connectors J7 and P2 and place them (do not solder yet) in the J7 footprint on Board 1 with the legs of the female connector J7 going through the board. Similarly, mate J8 and P3 and place them (do not solder yet) in the J8 footprint on Board 1.



Place your completed Board 2 from the previous

chapter on top of the assembly, component side up with the legs of P2 and P3 going through the corresponding footprints, and fasten the board to the 11mm standoffs with the two hex nuts. The resulting temporary assembly should be as shown in the photo.

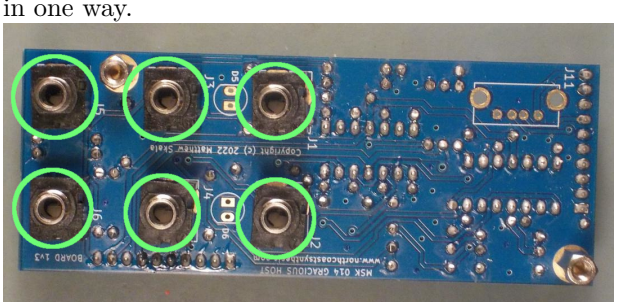


Solder J7, J8, P2, and P3 in place on the two boards. Then remove Board 2 and the hex nuts holding it in place, but keep the standoffs that go through the holes in Board 1.

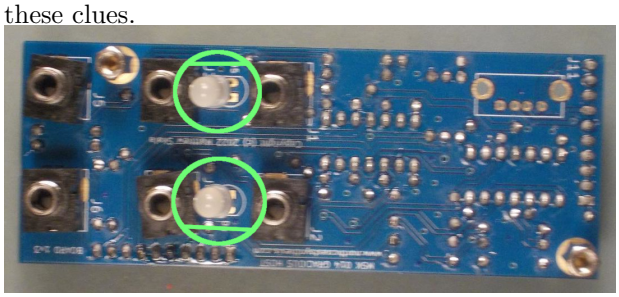
Panel components

Flip Board 1 over; you will now be installing the components that go between it and the panel. The pieces fit together in a straightforward way, but see the exploded assembly diagram on page 58 if further clarification is needed.

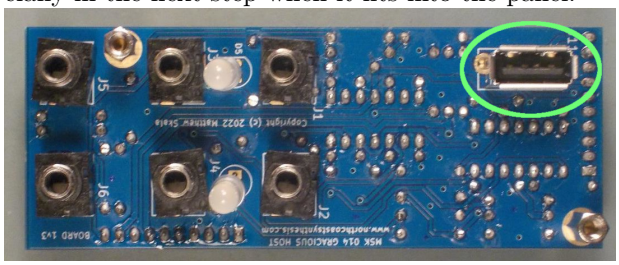
Place (do not solder yet) the six phone jack sockets J1 through J6 in their footprints. These are for patching signals to and from other modules. These components should only be able to fit into the board in one way.



Place (do not solder yet) the two bi-colour LEDs D5 and D6 in their footprints. Single LEDs are polarized and can be destroyed by reverse voltage. These ones here are special bi-colour devices with two separate LEDs in each package. The internal connection is such that each one protects the other from reverse voltage; so if connected backwards, they will not be destroyed, but the intended green and red colours will be swapped. Each LED lens has one flat side, and one leg shorter than the other on that side. The short leg is Pin 1. Its proper place on the board is marked by a circle with a flattened side matching the direction of the flattened side on the LED lens, and an oval solder pad. The other leg (Pin 2, long, farther from the flat side) goes into the rectangular solder pad. Be sure both LEDs are placed right way around according to these clues.



Place (do not solder yet) the USB A connector J11 in its footprint on the board. There should be only one orientation in which it will fit. The tabs on the back of the connector will snap into the mounting holes on the board, holding the connector in place; but you may need to adjust its angle slightly, especially in the next step when it fits into the panel.



Line up the panel on top of the asembly. Be sure that the top of the USB connector fits into the panel hole provided for it and is not trapped under the panel. Fasten the panel in place by driving the two machine screws through their corresponding holes into the 13mm standoffs.

Carefully flip the assembly over and let the jack sockets fall into their panel holes; the bushings should go all the way through the panel, with the legs just poking through the circuit board. Install the knurled nuts provided for the jack sockets, to hold them in place on the panel.

Do not overtighten any of this hardware, and be careful, if you are using wrenches or pliers, to avoid scratching the panel. Wrapping the tool jaws with tape may help.

Make sure that the LEDs poke through their corresponding panel holes by the amount you prefer. It may be necessary to push or pull on their legs at the back of the board in order to get the depth right.

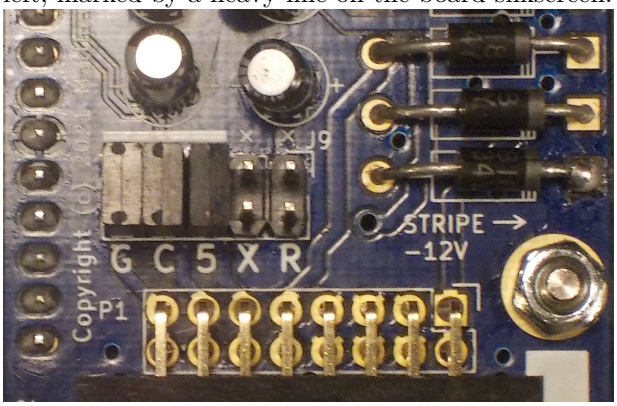
Solder all the panel components you just installed.

Final assembly

Insert the LM224 chips in their sockets on Board 1. Be careful to insert them right way round, with the Pin 1 markings on the chips matching those on the board, pointing down toward the bottom of the finished module.

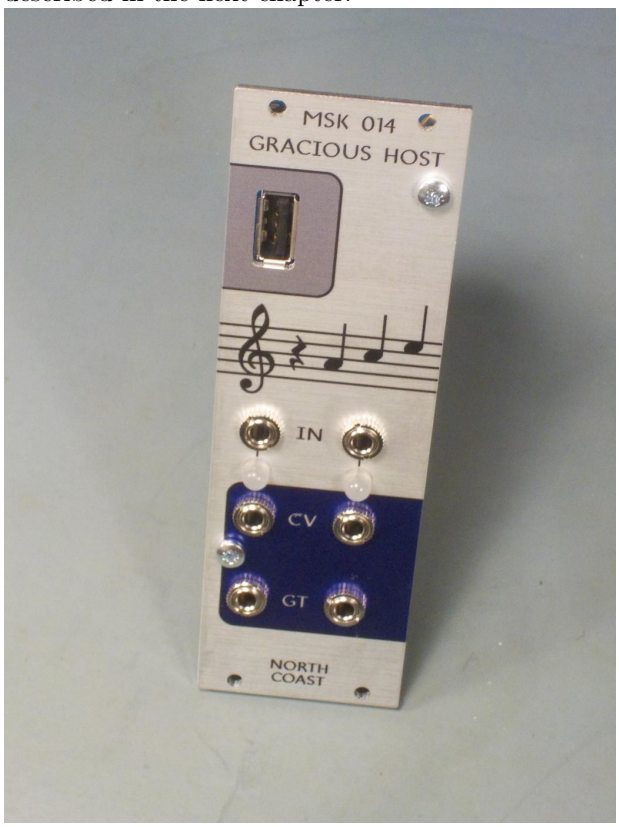
It should not be necessary to remove the panel from Board 1 again. Just attach Board 2, carefully fitting its header plugs into the header sockets on Board 1 and the male ends of the standoffs through the corresponding holes in Board 2. Then use the hex nuts to fasten Board 2 in place.

Install the configuration jumpers on J9, on Board 2. Their function is described in the "general notes" chapter of this manual. The recommended default configuration is to place jumpers in the locations marked G, C, and 5, which are the three locations at left, marked by a heavy line on the board silkscreen.



There is a rectangular white area at the right of Board 2 reserved for adding a serial number, signature, quality control marking, or similar. Use a fine-tipped permanent marker to write whatever you want there. Isopropyl alcohol will probably dissolve marker ink, so do this step after any board-cleaning.

The hardware of your module is complete at this point; but to get the voltages as accurate as possible, you may wish to run the calibration procedure described in the next chapter.



Firmware update and calibration

Many of the Gracious Host's features are defined by *firmware* – that is software built into the hardware – and it is possible to replace the firmware on an existing module with a new version. Before attempting a firmware update you should make sure you know *why* you are updating your firmware. If your module works and you are satisfied with it, you may not need to make any changes. The update process involves some risk and should not be attempted for no reason. But it is possible that I, or third parties, may release enhanced firmware with special features in the future. You may also have created new firmware of your own using the information in the *MSK 014 Gracious Host Programmer's Manual*.

After loading the firmware, the module also needs *calibration*, which adjusts the mapping between analog values in the outside world and digital values used in the firmware, to account for component values and module-to-module variation. The calibration process is mostly automated. It involves using a standard V/oct VCO as a reference.

If you have built your own module, it will need to be calibrated. Running the firmware update is one way to accomplish that, but you can also plug in a typing keyboard and enter maintenance code 5833, as described in the chapter on typing keyboards, to run the calibration procedure without updating firmware first.

To complete the update and calibration process, you will need:

- The Gracious Host module;
- a V/octave VCO that can track reasonably well over 0V–5V control voltage input;
- a couple of patch cables and the necessary power supply to run both modules at once;
- a USB flash drive and computer capable of writing files to it; and
- the firmware image file you want to use.

Preparing a firmware image

You will need an *image file*, which contains the executable firmware code in a special format specific to the Gracious Host. Up-to-date official firmware images are available from the North Coast Synthesis Web site at https://northcoastsynthesis.com/, as is the source code. Third-party firmware images may be available elsewhere.

The official firmware is subject to the GNU General Public License, and so must be any others that include parts of it. Among other things, that means third-party distributors of installable binary images are required to share their source code if their images contain any part of the original firmware.

The process of loading new firmware depends on code included in the *old* firmware. It is possible that loading a bad firmware image may "brick" your module, making it unable to load any further updates through USB. For that reason, you should be careful about loading unknown or untested firmware images. The module will not load an image file that fails a check of the CRC32 values included in the file, so mere file corruption is unlikely to result in a bricked module. But buggy code correctly packaged could possibly be a problem.

Prepare a USB flash drive in the format typically used by Windows. Most other popular operating systems can also format flash drives this way. The default settings should work. In more detail: the drive needs to have at most 2^{32} blocks, which translates to a maximum capacity of 2T if the blocks are standard 512-byte size; FAT12, FAT16, and FAT32 should all work, with FAT32 having received the most thorough testing; and the filesystem needs to be either written to the entire drive, or in a primary (not extended) partition described by a standard DOS partition table.

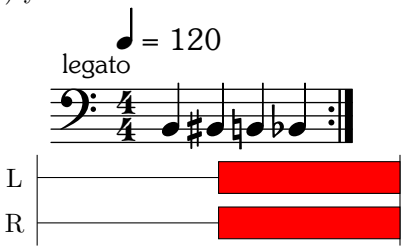
Rename the firmware to "FIRMWARE.FRM" if that is not already its name, and put the file in the root directory of the flash drive. The module will ignore all subdirectories and all files with other names.

Firmware update

Power up the module, and insert the flash drive. The module will automatically detect the drive, scan it for a valid firmware file, and attempt to reflash itself with the new image. That normally takes less than a

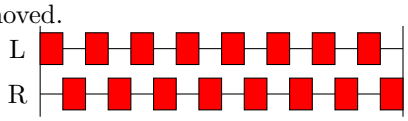
second. If the update succeeds, the module will start the calibration process, below, with slow-flashing red LEDs.

If firmware update fails, the module will perform the *failure display*, which consists of both LEDs flashing red slowly (1 Hz, 50% duty cycle) and an ominous tune played in 0V–9V square waves on the digital output jacks. If you patch either output jack to a speaker or headphone (beware of the DC offset; putting it through an AC-coupled module of some kind might help) you can hear this.



The failure display lasts one minute, after which (if you haven't powered it off already) the module will reboot. If you leave the flash drive inserted throughout this, then it will attempt the update again as soon as it does reboot.

If the module is unable to even attempt the firmware update – for instance, because it cannot interface with the flash drive – then instead of the slow-flash failure display it may flash the LEDs back and forth very fast, in red, until the flash drive is removed.



There is not a lot that can be done to debug a failed update. Possible causes might include:

- a badly-prepared firmware file, or one actually intended for some other module;
- something wrong with the transfer of the file – for instance, if you do not actually have the entire file;
- something wrong with the flash drive formatting or preparation, so that the module cannot *find* the file;
- an unusual, incompatible flash drive; or
- hardware problems (perhaps more likely in a DIY situation), especially affecting the microcontroller's communication with the SRAM chip.

In most plausible failure cases the module should retain its old firmware unchanged after a failed update attempt, though I cannot promise that in every case. Attempting the update again after a failure should be safe, but it will probably just fail again if nothing has changed.

Calibration: general remarks

The firmware needs to know what numbers to send to the DACs to produce specific output voltages, and what numbers it can expect from the ADCs when it receives specific input voltages. These are called output calibration and input calibration respectively. Although the firmware comes with reasonable guesses built in, the exactly correct numbers will be different on each individual module because of natural variations in the components used to build them.

The Gracious Host does not contain any very accurate voltage reference that could be used for calibration, but it *does* have an accurate clock crystal, so it can measure times and frequencies to a precision that is more than good enough for rock'n'roll. The calibration process takes advantage of that ability by using an external VCO to convert voltages into frequencies. Most users will have a VCO, because one of the main purposes of the Gracious Host is to control a VCO anyway. The Gracious Host output calibration sends different numbers to the DACs, accurately measures the resulting VCO frequencies, and infers the voltages it must be producing. Then once the output voltages are known, they can be used to calibrate the input voltages.

One consequence of doing things this way is that your Gracious Host's voltage calibration is only as good as that of the VCO used to do it. If the VCO's tracking is off, the Gracious Host's voltages will be off in the same way. However, if you intend to usually use the Gracious Host with *that same VCO*, at a single standard tuning setting, then matching the VCO's errors can be a virtue. The Gracious Host's output voltages will end up calibrated to whatever voltages that particular VCO needs to give in-tune pitches, compensating for the tracking error, because the calibration was to the VCO's frequencies rather than the voltages that a perfect VCO would require. The inaccuracy would only become an issue if you subsequently tried to use the Gracious Host with some other VCO, or with the VCO's tuning significantly different from where it was when you did the calibration.

The calibration routine is multi-threaded and the left and right sides work independently, so you can concentrate on just one side and do it completely before working on the other; you can do output on one

side, then the other, and then do input; or if you have two suitable VCOs you can do both sides at once.

Check the jumpers on the back of the Gracious Host before performaing calibration. If it is set to bus mastering mode (jumpers installed at "G" and "C") and plugged into a bus with other bus-mastering modules (such as one or more other Gracious Hosts with these jumpers installed, or possibly other modules that drive the CV/gate bus), then it and the other module or modules will conflict with each other and then the calibration will at best be inaccurate, and most likely fail entirely. It is safest to remove these two jumpers while performing calibration. Do not remove the "5" jumper unless you have modified your module to add an onboard regulator.

Output calibration

Calibration of either channel starts with the LED on that channel's side blinking red, in long slow pulses with short gaps between them (1.05 s overall cycle time).

When you see these blinks, any firmware update as such has already completed. It is safe to remove the USB drive at this point, and you should remove it before the end of the calibration process.

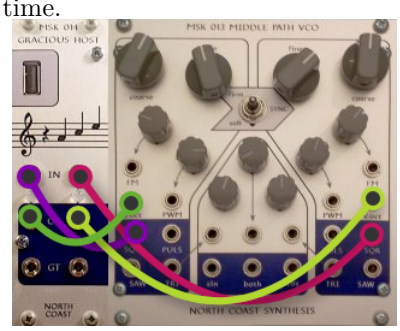
Tune your VCO so that at 0V input it will produce a frequency between 31 Hz and 277 Hz. If your VCO tracks well then the exact frequency does not matter much here, because the Gracious Host is only measuring the VCO's response to voltage, which ought to be the same across a wide frequency range. If your VCO does not track well and you are relying on the Gracious Host to correct the VCO's tracking errors, then you should tune it the way you normally will. For standard MIDI concert pitch that would ideally be 65.406 Hz at 0V (that is the C two octaves below Middle C), but the measurement range allows for tuning the VCO down an octave or up as much as two octaves relative to standard MIDI.

Turn off any sync, modulation, waveshaping, and similar features of the VCO.

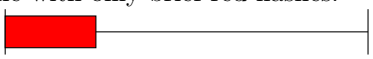
Patch the analog (CV) output from the Gracious Host channel you are calibrating, into the VCO's V/octave control voltage input. Do this before patching the VCO output.

Patch the VCO output into the input of the Gracious Host channel you are calibrating. Use the square wave output if the VCO has one; if not, use any simple waveform, such as sine, sawtooth, or triangle. The Gracious Host uses an input threshold of approximately +1.62V, and you want a waveform that will pass through that voltage once in the upward direction and once in the downward direction on each cycle.

This image shows both channels of a Gracious Host hooked up for output calibration using the two cores of an MSK 013 Middle Path VCO; but remember that it is not necessary to have a dual VCO. You could use a single VCO for both sides, doing them one at a time.



The module will automatically detect when a suitable VCO signal is present, and will start sending different control voltages and measuring the resulting frequencies. It is trying to adjust the voltages for ten different notes, while also re-confirming the frequency at 0V. As more of the voltages are detected as on-target, the red flashes get shorter while keeping the same repetition rate of one flash every 1.05 s. When nearly complete the LED will be off most of the time with only brief red flashes.

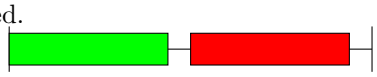


You can follow the progress of the output calibration by watching the length of the flashes. How long it takes depends on the VCO's stability and how far the previous calibration was from the new one, but it is normally expected to complete in less than a minute, and it may be only two or three seconds if the previous calibration almost perfectly matches the new one.

When output calibration completes the channel will wait to be repatched for input calibration.

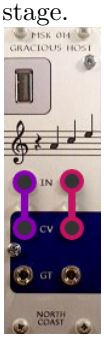
Input calibration

When the channel is ready for input calibration it switches to a faster blink pattern in alternating green and red.



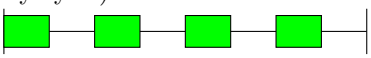
For this step, disconnect the VCO and just patch the Gracious Host's analog output directly to its own

analog input on the same side. This diagram, like the last one, shows both channels so patched, but remember that the channels are independent and you can do input calibration on one of them while the other is at a different stage.

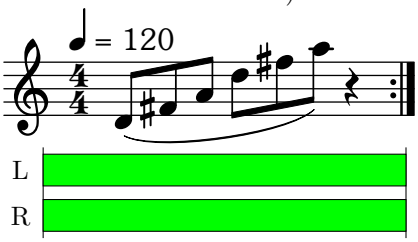


In principle, the input calibration will shorten its LED blinks to indicate progress, in the same way that the output calibration did. However, the input calibration is usually so fast (two or three seconds per channel) that you may not see the shorter blinks before it finishes.

When one channel has completed input calibration, it will display quite fast green blinks (3.8 Hz, 50% duty cycle) while it waits for the other.



When *both* channels complete input calibration, both LEDs will go solid green as part of the *success display*, and the digital output jacks will play a happy tune in 0V–9V square waves. You can patch either output jack to a speaker or headphone (remaining aware of the 4.5V DC offset) to hear this.



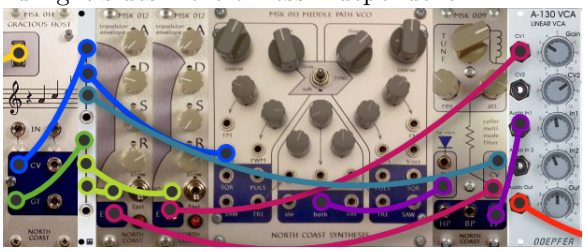
The success display lasts one minute, after which the module will reboot.

If you initiated calibration by doing a firmware update, then you ought to have already disconnected the USB drive before the end of the success display; otherwise, on reboot it will attempt to update the firmware again.

The calibration data is written to the module's non-volatile memory at the *start* of the success display, so it is safe to power down the module as soon as you see the solid green LEDs. If you wish to abort calibration once started, just power the module down before reaching the end. In that case, the module will end up with the default values from the firmware you loaded, if you loaded firmware; or no change to the existing calibration if you started calibration in some other way, such as with maintenance code 5833.

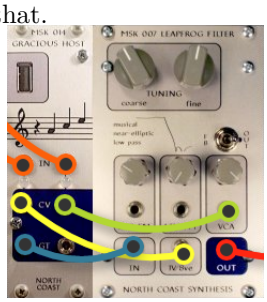
Patch ideas

Here's a straightforward subtractive patch controlled by the left pitch CV and gate. Pitch CV is split by a multiple to control the Middle Path oscillator (both sides, through normalling in that module) and the Coiler VCF. Gate CV is also split to control two ADSR envelopes, for the VCF (second CV input) and the VCA. This patch would work well with a MIDI keyboard or a mouse plugged into the Gracious Host. Using MIDI channel 2, the combination of channels 8 and 9, or the two buttons of a mouse, this patch could be expanded to control modular voices, with the left and right sides more or less independent.



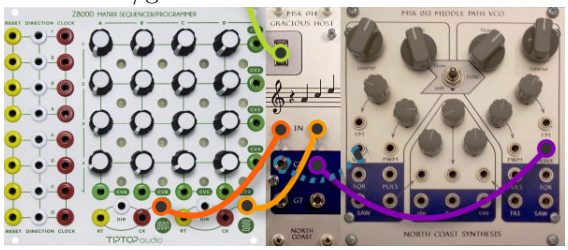
The next patch shows the "non-USB" behaviour of the Gracious Host: with nothing plugged into the USB port, it acts as a primitive oscillator and envelope, taking pitch and gate CV as standard Eurorack control voltages and producing a fixed-parameter ADSR envelope (upper right output jack) and square wave oscillator output (lower left). Here the Leapfrog VCF, with its built-in VCA, is providing the rest of the patch (filtering and articulation). The quantized pitch CV output from the Gracious Host is shown being used as the V/oct input for the filter to save the need for a multiple, but it would also make sense to patch the original (unquantized) pitch CV to the filter instead.

This patch as shown will have a significant DC offset on its output; any AC-coupled module, such as the AC output of the MSK 011 Transistor Mixer, could remove that.



Here is a minimal patch intended to be controlled by a MIDI or typing keyboard on MIDI channel 1, using the Gracious Host's built-in oscillator feature. The pitch CV controls the filter, the gate CV activates an ADSR envelope, and the velocity CV goes to the filter's second CV input for additional expressive control. Another option might be to patch it to the VCA to vary the volume.

This patch (really a partial patch) shows how the Gracious Host might be put between a sequencer and an oscillator for MIDI-controlled quantization. Play several notes on MIDI channel 3 or 4 and the notes from the sequencer will be shifted to the nearest MIDI notes; so a single sequence can play in multiple keys or scales under keyboard control. The left pitch CV is shown dotted because this cable does not actually need to be patched; the Gracious Host can send, and the Middle Path can receive, control voltages over the Eurorack CV/gate bus.



A USB mouse can be a general-purpose CV controller, not only used to play notes. In this patch it controls two volume levels with vertical and horizontal movement, for mixing and cross-fading between two sampler modules. The gate outputs, shown unpatched, could be connected in a more complicated patch to switch between samples.

Circuit explanation

The Gracious Host is a digital module and most of its features are implemented in firmware. There is another entire volume, the *MSK 014 Gracious Host Programmer's Manual*, describing how the firmware works. That leaves very little to be said about the hardware as such. But here are a few comments about the circuit, especially concentrating on the analog parts.

Digital parts and power

Most of the intelligence in the module is inside a Microchip PIC24FJ64GB002 microcontroller chip. This is basically a complete 16-bit microcomputer on a chip with 8K of built-in RAM, a bunch of peripheral devices, and 64K of built-in flash memory used to store the firmware. Kits and built modules come with preprogrammed chips (possibly not the very latest version of the firmware); new firmware can be loaded over the USB port.

There are a few other digital components directly wired to the relevant ports on the microcontroller: a 4.000 MHz clock oscillator; an MCP4822 12-bit two-channel DAC; and a 23LC1024 serial RAM with a capacity of 128K. The oscillator provides accurate timing control, which is especially required for USB communication. The DAC controls the analog output jack voltages. The SRAM chip, in the current firmware, is used as a buffer during firmware upgrades; it is also available for use by future and third-party firmware which may need more than the microcontroller's built-in 8K RAM.

Pin 14 of the microcontroller is brought out to the test point P5 on the back of the module. With the standard build of the firmware, there is no signal on this test point, but debugging code can send signals to this point where they can be observed with an oscilloscope.

None of these digital components requires much in the way of support circuitry beyond decoupling capacitors, including the extreme 10 μF ceramic cap on pin 20 of the microcontroller. The digital components all run on +3.3V power, regulated down from the Eurorack +5V supply by an MCP1700-3302 LDO regulator. The microcontroller actually uses a lower voltage internally; that large capacitor is intended to support the microcontroller's built-in regulator, which seems to be a switching type for reasons of power economy. The microcontroller also has a connection to the +5V supply, to support the USB voltage requirements.

The power input is also straightforward. The ±12V rails have series Schottky diodes to protect against reverse connection, although this is less an issue on the Gracious Host with its 16-pin connector than would be the case on a module with a 10-pin power connector. Reversing the 16-pin connector shorts the +5V and +12V lines into ground and will probably cause the power supply to shut down relatively harmlessly. The power rails also have 10 μF electrolytics to reduce general interference, though the smaller per-chip decoupling caps will have a more significant effect. The +5V power rail has no Schottky diode because we cannot afford the voltage drop, but it does go through a 200mA polyfuse, with a 150 μF capacitor on the far side, according to Microchip's recommended circuit for powering the USB bus.

Except for the connections to the USB port, which have their own built-in drivers and protective circuits, all the digital components are buffered from the outside world by op amp circuits running on the Eurorack ±12V supply, which translate the levels and protect the digital components from out-of-range voltages. Those are discussed in more detail in the next few sections.

Analog input buffers

Figure 6 shows the circuit for one of the input jacks. The same section is repeated for the other.

This circuit is intended to serve two goals: first, map the input range 0–5V (intended useful range of the input jacks) to be within 0–3.3V (the measurement range of the microcontroller's built-in ADC); and second, make sure that no voltage within ±12V applied to the input jack can result in a voltage applied to the microcontroller outside the range −0.3V

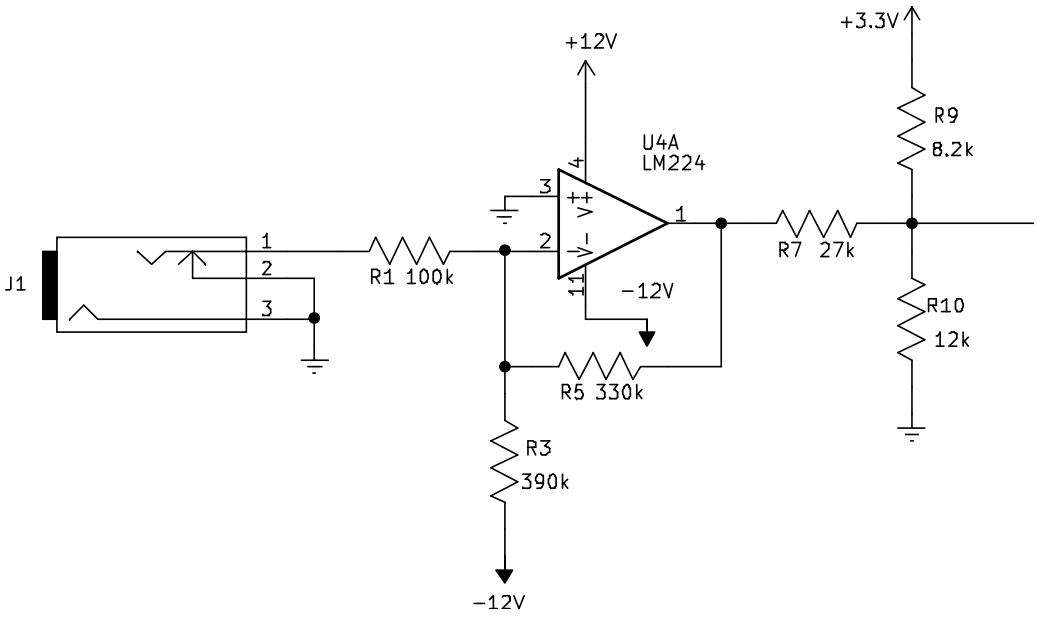


Figure 6: Analog input buffer.

to +3.6V, representing the microcontroller's absolute maximum rating. It ought to achieve these goals in all conditions of component variation within tolerance. It also ought to present not too large an impedance to the microcontroller input, because ADC conversion time suffers when the driving impedance is too large.

The circuit topology is best understood working back from the microcontroller input. We could imagine that the op amp output might vary over as much as ±12V – not really, because this op amp is not rail to rail, but that is a conservative worst case. No matter what goes on at the input jack, we want the op amp to be unable to apply a damaging voltage to the microcontroller. The three-way voltage divider of R7, R9, and R10 is designed to guarantee that no op amp output (divider input) voltage inside ±12V can result in a microcontroller input (divider output) voltage outside −0.3 to +3.6V. Also, the output impedance of the divider (which will be equal to the parallel combination of the three resistors) ought to be no more than about 5kΩ, for compatibility with the ADC's drive requirements.

I selected the divider resistor values mostly by trial and error with a simulator. But a rough calculation to guide such experiments proceeds by simplifying the circuit: instead of a three-resistor divider between the op amp at ±12 and the 0V and 3.3V rails, we can think of it as a two-resistor divider between the op amp and *some unspecified constant voltage to be chosen later*. The resistors R9 and R10 are replaced by their equivalent circuit. Then it's easier to analyse. One side of the divider is R7, and the other is the parallel combination of R9 and R10 (R9‖R10), which I will call RX.

Knowing that we need the 24V input range to map to at most a 3.9V output range means the gain through this divider can be at most 3.9/24, about 0.162; and then R7/RX should be at least about $(1 - 0.162)/0.162 = 5.17$. Combining that with the target of R7‖RX=5kΩ (desired output impedance) we can write out the equations and solve them to get starting estimates of R7=30.85kΩ and RX=5.97kΩ. None of this depends on knowing the constant voltage on the other side of the two-resistor divider.

The next step is to get a target for that constant voltage. If we decide +12V input should map to +3.6V output, then Ohm's law with the tentative value of R7 says that the 30.85kΩ resistor, dropping 12V−3.6V=8.4V, will carry 272μA. That much current through RX with the tentative value of 5.97kΩ gives a voltage drop across RX of 1.62V from the 3.6V output value, and the other end of RX is at 1.98V.

Now we choose values for R9 and R10, connected to +3.3V and 0V respectively, such that their combination will be equivalent to a sin-

gle 5.97kΩ resistor connected to 1.98V. Dividing 1.98/3.30 = 0.600 indicates that R10/(R9 + R10) should be close to 0.600. Combining that with R9‖R10=RX=5.97kΩ and solving gives estimates R9=9.95kΩ, R10=14.92kΩ. Those and the earlier value R7=30.85kΩ are reasonable starting points for simulator experiments. I started with standard values near there, simulated the input/output function in Qucs, and tweaked the values until I got results that looked reasonable. I ended up with all three resistor values a little smaller than these estimates, presenting a slightly lower impedance to the ADC.

Choosing the resistors around the op amp (R1, R3, and R5) was simpler because of fewer interactions between them. Wanting to have a typical Eurorack input impedance of 100kΩ dictated the value of R1 as 100kΩ. Although I calculated the divider downstream on the assumption that the op amp might produce any voltage in ±12V, really I only think I can reliably expect it to swing over ±10.3V. That is a 12V rail on each side, minus 0.2V eaten by the reverse-protection diode, minus 1.5V to reflect the op amp's output swing capabilities (probably better than that on the negative side but this is the specification for the positive side and I am conservatively assuming the same on both sides). So this circuit should be free of clipping on a 5V input range if it maps it to a 20.6V output range, and that means the magnitude of the amplifier's negative gain should be no more than 20.6V/5V=4.12; R5 should be no more than 412kΩ, and probably near that much. Then R3 should be whatever it takes to shift the 0–5V input range to something that, when seen through the downstream voltage divider, will be comfortably within the microcontroller's 0–3.3V measurement range.

I did not bother to calculate a precise estimate for R3 but went directly to the simulator and tried different standard values for it and R5 until I found some that seemed to work well. The op amp gain ended up smaller than estimated above partly because 20.6V is actually a little too wide as the output range from the op amp: the downstream voltage divider is designed to map all possible op amp voltages to a *non-damaging* range of 3.9V, but we need 0–5V at the input jack to map to the *measurable* range of 3.3V, and do it in all possible conditions of component tolerances, and given the use of convenient standard values for the components instead of exactly calculated ones. There is also some interaction and compromise between the offsets introduced by R3 and by the R9/R10 voltage divider. Playing with different standard values until all the voltages worked out in simulation eventually gave the choices shown on the schematic.

LED drivers

The LED driver section is as shown in Figure 7. It is repeated on both sides, sharing the voltage divider of R13 and R14.

This circuit is driven by GPIO pins on the microcontroller, which have three states: high, +3.3V and able to source a lot of current; low, 0V and able to sink a lot of current; and high-impedance or tri-state, in which the pin effectively disconnects from the circuit entirely, neither sourcing nor sinking current. The LED is a bi-colour unit, actually two LEDs wired anti-parallel, so current in one direction will make it light up red, and in the other direction, green.

We want to drive a controlled amount of current through the LED in each direction, so that the brightness of each colour will be roughly equal. This is complicated a little by the fact that the LED will have different luminous efficiency (brightness per milliamp) in each direction, and the *voltage* it takes to drive that current will vary with the direction, individual LED unit, colour, temperature, and ambient light level, at least.

Just connecting the LED between the GPIO pin and some constant voltage in the middle of the 3.3V range would work if we could depend on consistent forward voltages for the two colours, adding up to 3.3V. Adding a series resistor would help even out variations in the forward voltages, if we could depend on the sum of LED forward voltages being significantly *less* than 3.3V. But in fact, the forward voltages needed for red and green LEDs are usually more like 2V each, so the microcontroller's output driver with a 3.3V swing cannot light the LED in both directions with such a simple circuit. Putting the LED between two GPIO pins, with a resistor to help control the current, would double the voltage swing and might work but would require more pins than are available. There would also be issues with controlling the current in the two directions separately. So the Gracious Host instead uses an op amp circuit that controls the current more precisely and is capable of operating over a wider voltage range. The driver basically uses the "connect GPIO pin to a constant voltage in the middle of the range" approach but it puts a well-behaved resistor there instead of the LED, allowing the use of lower voltages, and then it echoes the current through the resistor into the LED.

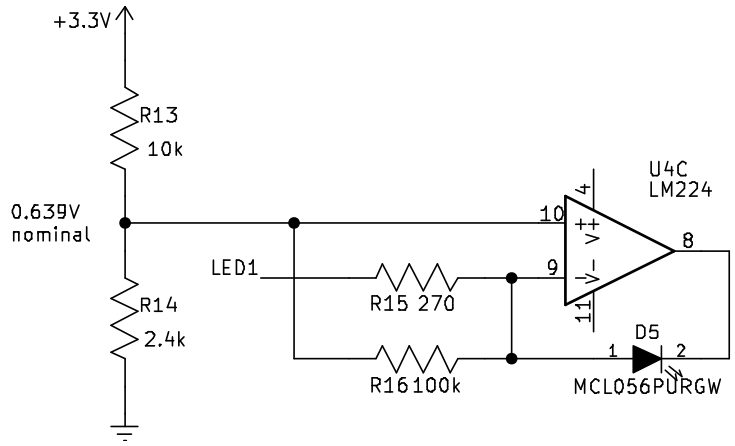


Figure 7: LED driver.

The resistor R15 is really the heart of the circuit: it converts voltage to current. Op amps in negative feedback always act to bring their inputs to equal voltage. The R13/R14 voltage divider provides a basically constant voltage (about 0.64V) to the positive input of the op amp, and then we can assume that the negative input will be at 0.64V too. When the GPIO pin is high (3.3V), the voltage drop across R15 is 3.30V−0.64V=2.66V and the current through R15 is 2.66V/270Ω =9.85mA. When the GPIO pin is low (0V), the voltage drop across R15 is 0.64V *in the other direction*. The current through R15 is −0.64V/270Ω = −2.37mA.

These amounts of current flow through R15; but the total current into and out of the op amp's negative input must be zero (Kirchoff's Law and the high impedance of the op amp input). Effectively no current flows through R16 because of its high resistance compared to the components around it. So all the current through R15 must also be passing through the LED. High and low states of the microcontroller GPIO pin correspond to currents of +9.85mA and −2.37mA through the LED. The op amp will automatically adjust its output voltage to get these currents, regardless of variations in the LED's forward voltage.

The overall brightness of the LED is determined by the value of R15: the sum of (the absolute values of) the two currents is necessarily 3.3V/R15, so we could make both colours brighter or dimmer by changing this resistor. The ratio of current for red and green is determined by the 0.64V voltage applied to the op amp positive input, and therefore by the ratio of R13 and R14. Raising or lowering this voltage increases current in one colour state and decreases it in the other. The actual values I chose were determined by experiment, starting with an estimate from the LED's data sheet values for visible brightness in millicandelas at different current levels. LEDs within their normal operating range tend to be quite linear in brightness relative to current.

But what about the high-impedance GPIO state, and R16? When the GPIO pin goes into the high-impedance state, we want the LED to turn off. In that case the GPIO pin is effectively disconnected, and so is R15. There is no current through R15 and the LED driver circuit, trying to match the R15 current through the LED as well, will basically put no current through the LED – as desired.

But although the current through the op amp's input is usually approximated as zero, it isn't really. Op amps always require a small *bias current* through their inputs, and for the LM224 in particular the bias current is on the order of 20nA, a small amount of current in absolute terms but actually relatively large for an op amp. Bias current generally depends on the type of op amp; bipolar-input units like the LM224 tend to have more than other designs. Some CMOS op amps designed specifically for ultra-low bias current are able to achieve orders of magnitude less. Without R16, the op amp would try to bring its negative input to the same voltage as the 0.64V at the positive input, by driving the tiny bias current through D5. The bias current would not be enough to light the LED.

The problem is that LEDs tend to be unpredictable at such low current levels, and op amps like this one are not really designed to produce tightly-

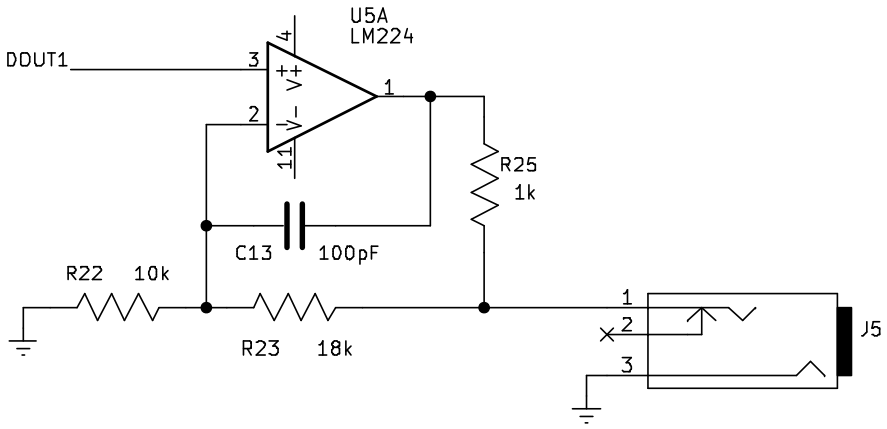


Figure 8: Output buffer.

controlled very low current levels either. The LED may end up behaving more like a capacitor than a resistor; it may also generate a small voltage of its own by converting ambient light into electricity, in an effect similar to what a solar cell does. What the op amp needs to do to drive 20nA through the LED will vary moment by moment and it may end up being unstable. In practice it would *probably* work fine anyway, but I wanted to be on the safe side.

To make sure everything stays under control in the LED-off situation, the resistor R16 ties the op amp inputs together, providing a path for the bias current to the relatively low-impedance output of the R13/R14 voltage divider instead of only through the LED. With the LED on, the current through the 270Ω resistor overwhelms the contribution of R16, which distorts the current levels from their target values only by a tiny amount. But with the LED off, the current through R16 overwhelms the unstable contribution of whatever comes through the even higher impedance of the LED. Limiting the impedance at the negative input to 100kΩ means the op amp can drive its output, and the far side of the LED, to 0.64V without needing to operate in the very low current range where it may not function well.

Output buffers

The output driver section is shown in Figure 8. All four output jacks use this circuit. The digital output buffers take their input from microcontroller GPIO pins, and the analog output buffers take their input from the DAC outputs. The GPIO pins have low and high states of 0V and 3.3V, which the output buffer circuit amplifies to 0V and a little over 9V. The DAC outputs (in the mode normally used) have a range of 0V to 2.048V, which the output buffer amplifies to a nominal range of 0V to 5V with a little extra margin available at the top (calculated limit 5.73V).

This is a reasonably straightforward non-inverting op amp amplifier circuit with a voltage gain of 2.8 and a couple of components added to improve stability. Ignoring R25 and C13 for the moment, R22 and R23 form a voltage divider that divides the output voltage as seen at the jack, by 2.8. The op amp in negative feedback always acts to make its inputs equal. So for any given voltage coming into this section along the wire labelled DOUT1, the op amp will bring its own output voltage to whatever voltage causes the output jack to be at 2.8 times the DOUT1 voltage. As long as the op amp is capable of doing that, the section has an overall gain of 2.8. Input of 0V gives output of 0V; input of 3.3V gives output of 9.24V; and input of 2.05V gives output of 5.73V.

But there can be some issues with a very simple op amp circuit, usually related to what happens if the user plugs something weird into the jack socket. In particular, if the op amp's output is *directly* connected to the jack socket and the user plugs in a short circuit, or another output similarly designed, then the op amp may attempt to drive the jack to a voltage that the external connection will not allow. The op amp will just dump more and more current into the conection, trying to raise the voltage. In principle, that kind of overload can be damaging. This particular op amp has built-in protection against such situations, so it probably will not damage *itself*, but the high current could be damaging to whatever is on the other end of that cable, and it could indirectly

cause problems by overloading power supplies, etc. Adding R25 limits how much current the op amp will be able to source or sink through the jack in an overload situation.

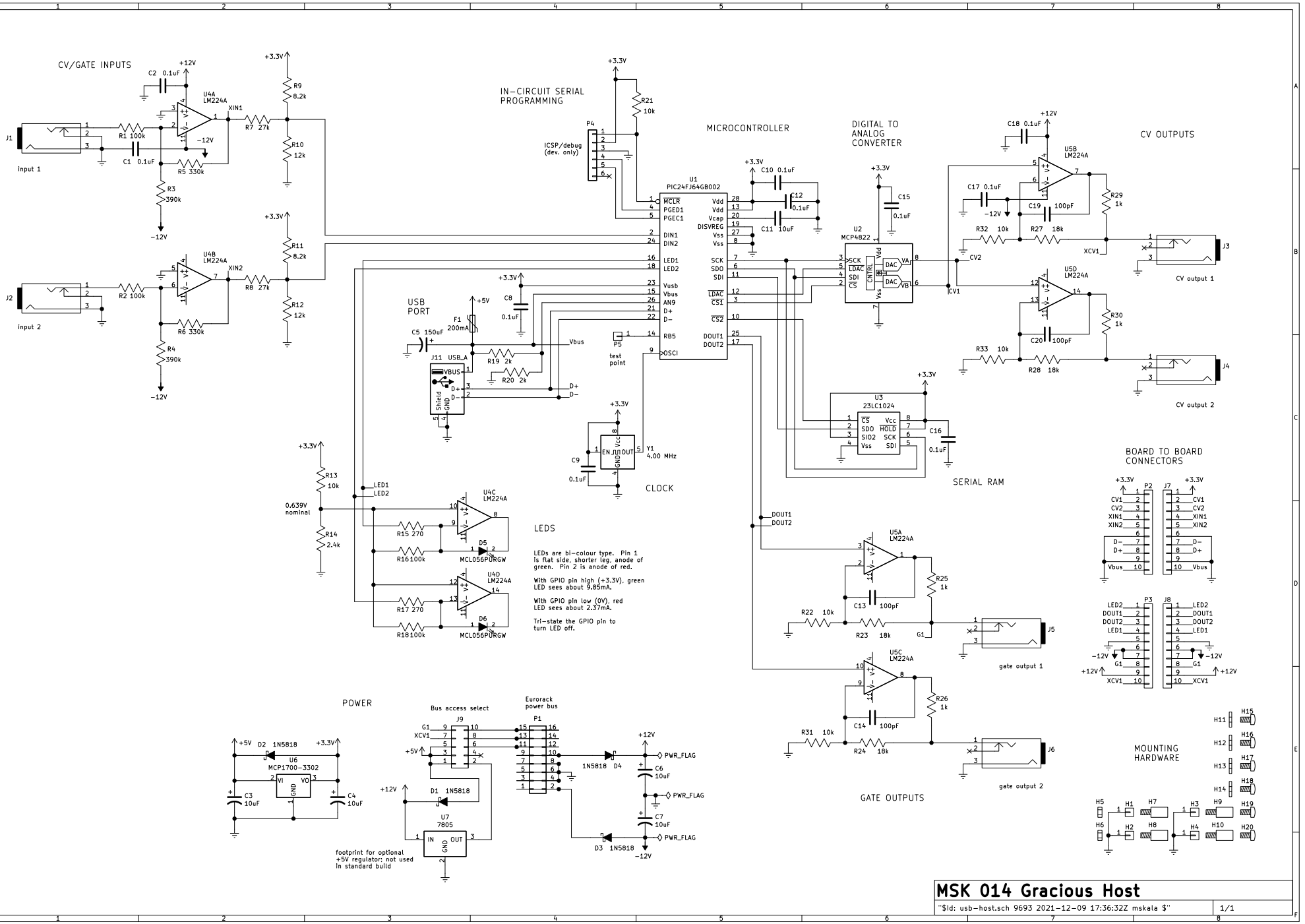
Another, subtler, issue has to do with stability. Suppose the user plugs something into the jack socket that exhibits significant capacitance. A very long patch cable would be an example. Capacitive loads are in a sense slow to respond to current. If the input goes high and the op amp tries to raise the voltage to match, the current flowing into the cable capacitance may require some nonzero time to charge the capacitance and really raise the voltage. So the op amp continues trying to raise the voltage even after the point at which it should have turned off its output current. Then the voltage ends up too high when things start to settle down, and so the op amp is forced to draw current in the other direction to lower the voltage, and it can overshoot there as well. Mild overshoots that die down after a few cycles are called *ringing* and result in the voltage bouncing around a little before settling down, every time it changes. In more extreme cases, a larger amount of cable capacitance can set up a self-sustaining *parasitic oscillation*, in which the op amp produces a continuously changing voltage that never settles down, usually over its entire output range with a frequency in the hundreds of kHz to a few MHz.

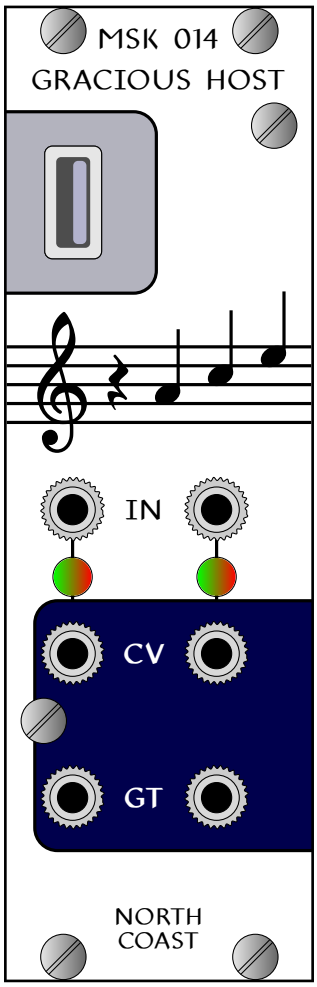
The resistor R25 helps prevent ringing and parasitic oscillation by limiting how strongly the op amp can apply current to the external connection, and therefore the extent of a possible quick overshoot. The capacitor C13 also helps, by inhibiting the op amp's gain at high frequencies. Capacitors have lower impedance as the frequency goes higher, so at the high frequencies associated with these instability phenomena, C13 applies extra negative feedback, reducing the gain below the level where temporary or sustained oscillation would be possible. At lower frequencies, the impedance of C13 is so high as to have little or no effect on circuit operation. The effect of C13 can also be stated in terms of its applying a phase shift that keeps the circuit away from the region on a phase and gain versus frequency plot where oscillation would be an issue.

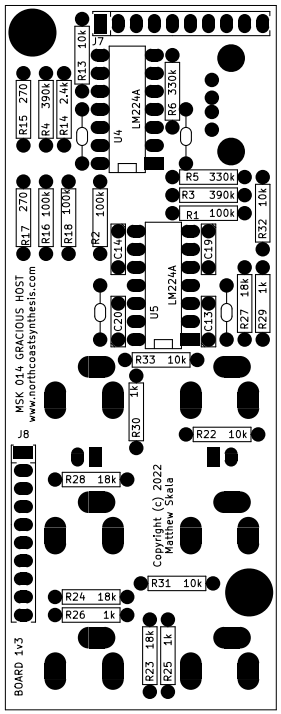
Mechanical drawings

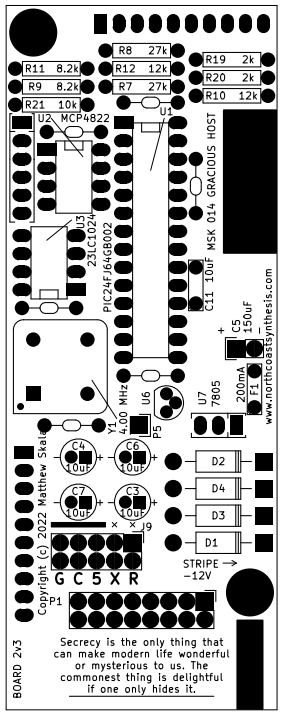
On the following pages you will find:

- the schematic diagram for the module;
- a mock-up of what the completed module looks like from the front panel;
- the top-side silk screen art showing component placement;
- the bottom-side silk screen art showing component placement (*note this drawing is mirrored, and shows what you actually see looking at the board, not the X-ray view used in other Kicad output*);
- a drawing of the front panel, with the hole locations and other information for manufacturing it; and
- an exploded isometric drawing showing how the boards and hardware fit together.


CV/GATE INPUTS
IN-CIRCUIT SERIAL PROGRAMMING
MICROCONTROLLER
DIGITAL TO ANALOG CONVERTER
CV OUTPUTS
USB PORT
CLOCK
SERIAL RAM
BOARD TO BOARD CONNECTORS
LEDS
LEDs are bi-colour type. Pin 1 is flat side, shorter leg, anode of green. Pin 2 is anode of red.
With GPIO pin high (+3.3V), green LED sees about 9.85mA.
With GPIO pin low (0V), red LED sees about 2.37mA.
Tri-state the GPIO pin to turn LED off.
POWER
GATE OUTPUTS
MOUNTING HARDWARE
footprint for optional +5V regulator; not used in standard build
MSK 014 Gracious Host
"$Id: usb-host.sch 9693 2021-12-09 17:36:32Z mskala $"
1/1

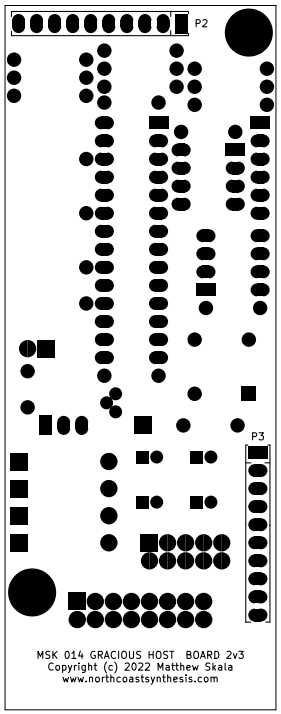

MSK 014
GRACIOUS HOST
IN
CV
GT
NORTH
COAST

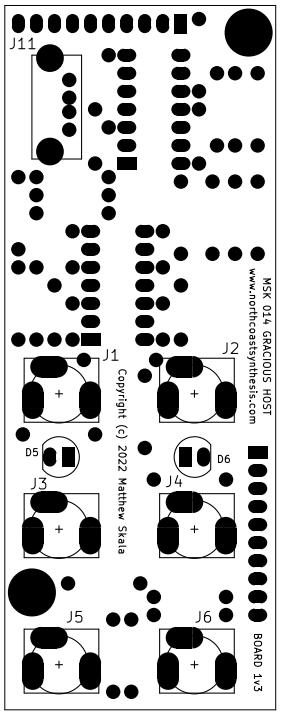




MSK 014 Gracious Host

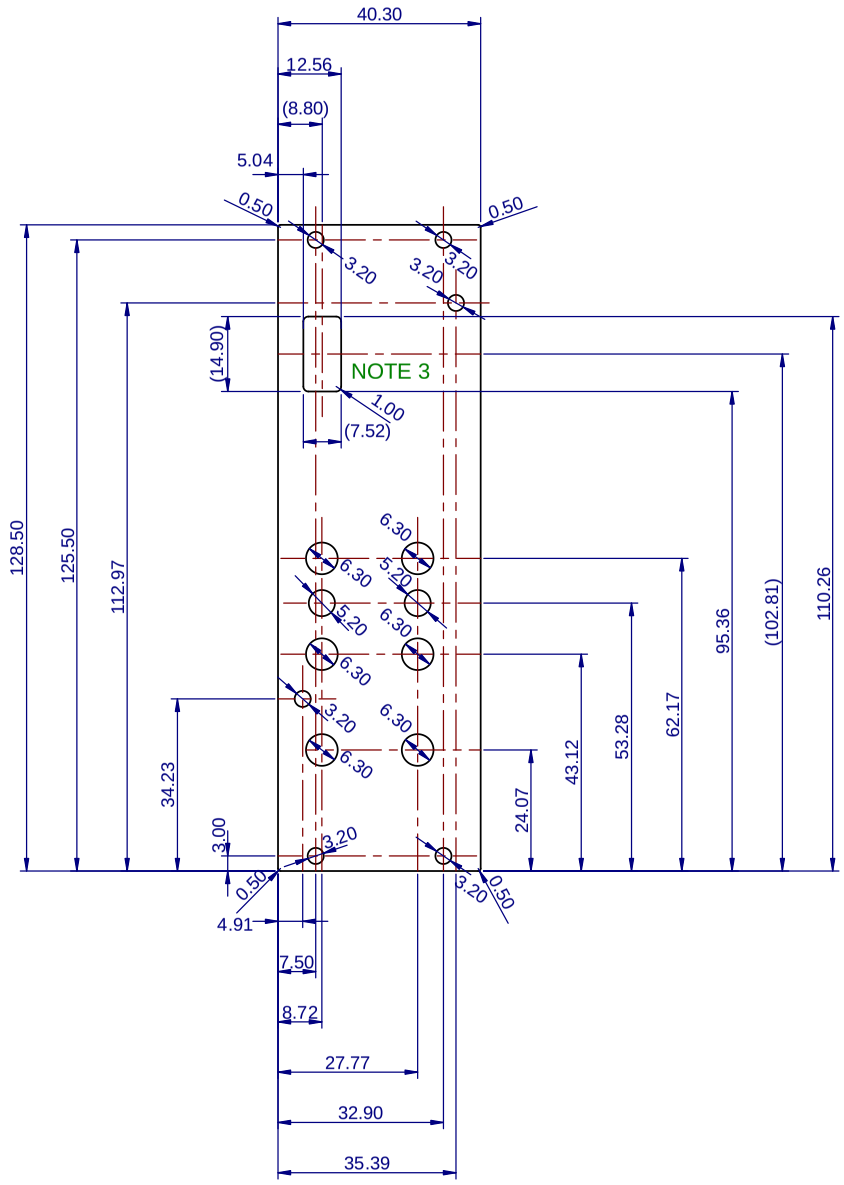
$Id: usb-host.kicad_pcb 9730 2021-12-21 20:25:11Z mskala $	1/1





MSK 014 Gracious Host

$Id: usb-host.kicad_pcb 9730 2021-12-21 20:25:11Z mskala $	1/1



NOTES

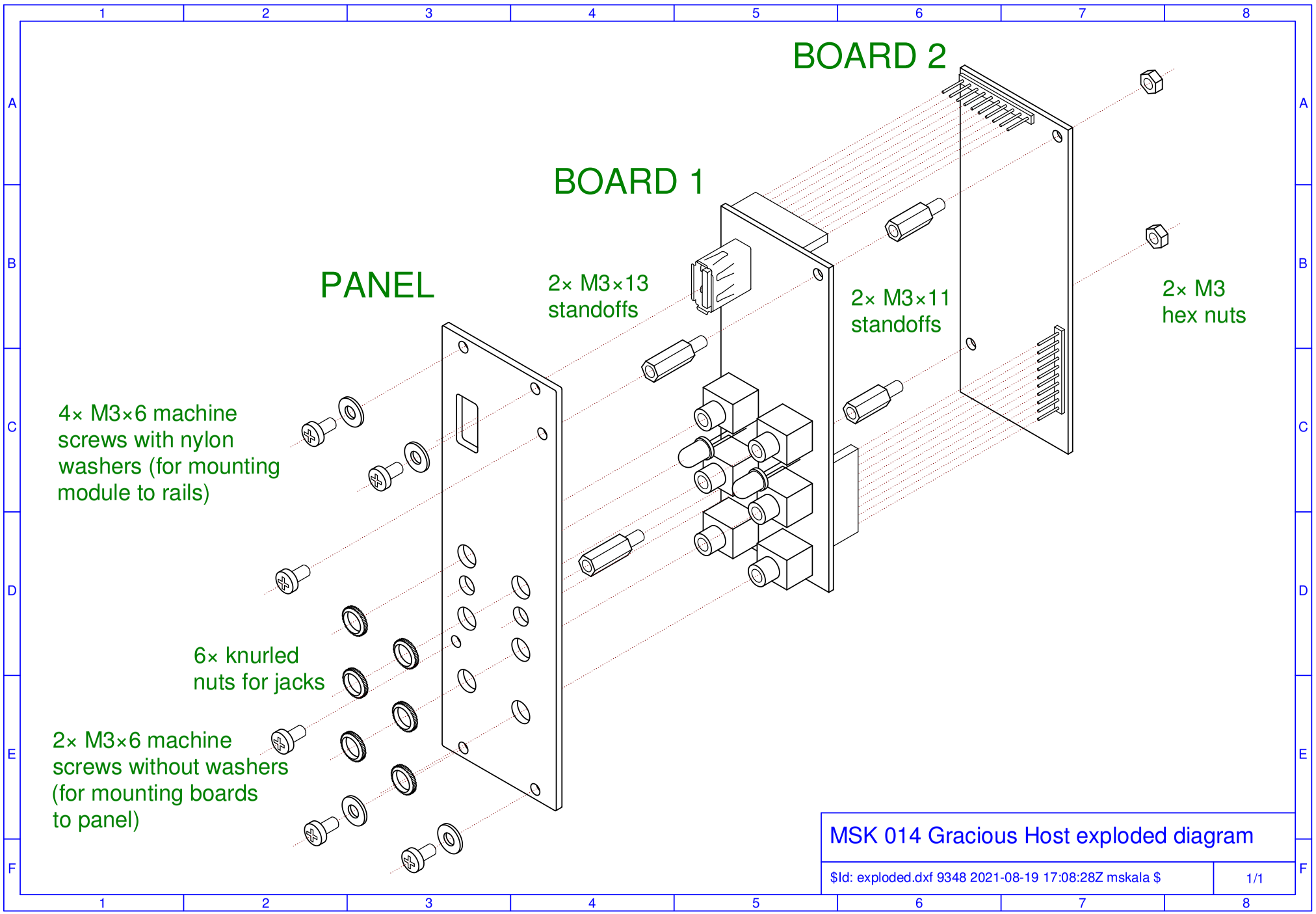
1: All dimensions are millimetres with tolerance +/- 0.10mm. Do not scale from drawing.

2: Material is aluminum, 2.00mm thick. Overall dimensions are 128.50mm by 40.30mm, per Doepfer spec for an 8HP Eurorack synthesizer module.

3: Rectangular hole for USB-A connector with 1.00mm radius all four corners. Parenthesized dimensions hole size and centre location for reference only. Do not confuse USB-A horizontal location with centre line of jack sockets below, which is 0.08mm further to the left.

MSK 014 Gracious Host panel mechanical

$Id: panel-mechanical.dxf 9383 2021-09-03 21:36:22Z mskala $ 1/1



MSK 014 Gracious Host exploded diagram

$Id: exploded.dxf 9348 2021-08-19 17:08:28Z mskala $