Foenix Rising
A newsletter for developers, users, and enthusiasts of Foenix Retro Systems products
SIDlab beta 2
beta 2

Welcome to SIDlab, a Foenix Rising production

SIDlab is a playground for experimentation with your physical MOS6581, MOS8580, pin-compatible replacement SID (ARMSID, BackSID, SwinSID), or the Gideon SID built into your F256K. A few planned features are absent from this beta version, but there is plenty to keep your busy.

SIDlab is written in 65C02 assembly language (65C816 run-able) and is packaged with sample song data in a .PGX executable. Additional music and features are planned for the prod release (post April/May 2024).

Origins

SIDlab began with a few dozen lines of code ported from the Commodore "Christmas Album", and has evolved since initially demonstrated as 'guru mode' of the FoeniXmas23 demo.

Originally, a feature was added to produce a hex dump (similar to the screenshot on pg. 2) and a method to pause and single-step the play-engine for observation and learning. Since being branded *SIDlab*, it now supports a keyboard interface, modification of all of the parameters of the SID chip, and more.

There is an accompanying 12-episode YouTube series that demonstrates the app, entitled: *The 12 Days of Commodore's "Christmas Album"* also known as *12DoC and the Commodore SID chip.* If you are not aware of the series, have a look (episodes 7 and 8 are squarely focused on SIDlab).

Target audience and purpose

SIDlab is intended for anyone who wants to learn more about the legendary Commodore MOS integrated circuit and its capabilities, or for anybody that is working on an F256 (or Commodore 64/128) application and needs a workbench to test a sonic 'what if' scenario with ease. In the short term (this release), SIDlab can be used to model a patch or sound effect, and will dump parameters to the screen for use in your program; longer term, patches can be saved to disk. SIDlab is not a tracker or a general purpose SID player, but it has a place in this ecosystem.

Historically, the SID chip has been a bit of a beast to work with, partly because of its power and complexity, but also, due to a lack of tools. SIDlab intends to change this and in doing so, aims to improve upon the beeps and blips normally associated with computer audio. Its strong suit is connecting analog synthesizer concepts to capabilities of the classic circuit. In time, it will be ported to other platforms including the Foenix GEN-X, NitroOS-9 on the FNX6809, and potentially, over to the Commodore 64 and 128.

Music Support and the on-screen keyboard

For now, the only songs supported in SIDlab are the 7 Christmas Demo songs that may still be ringing in your ears. But rest assured, the ability to mute and drastically alter voice attributes, help alleviate the reality that the holiday has past. And to help, Johann Sebastian Bach's Invention #13 has been moved to the first song in the queue. Longer term, the ability to load other SID tracks from a vast library will be supported.

SIDlab includes a rudimentary method to *test play* your patch as it is being developed (pressing `ctrl`-'1', `ctrl`-'2' or `ctrl`-'3' will play a voice with the selected parameters and `ctrl`-'a' will play all in unison). The user may also use the computer keyboard (see the map on pg. 5) to play notes on a mini-keyboard. It's basic, but it works, and you can exercise left and right hand keys independently*. In a future release, right-hand playing will permit stacking of two voices in 3rds, 5ths, slightly detuned, or +1 / -1 octave intervals.

* due to keyboard matrix intricacies, full key independence may be limited to PS/2 keyboards; discussed on pg. 3 below

Execution and Startup

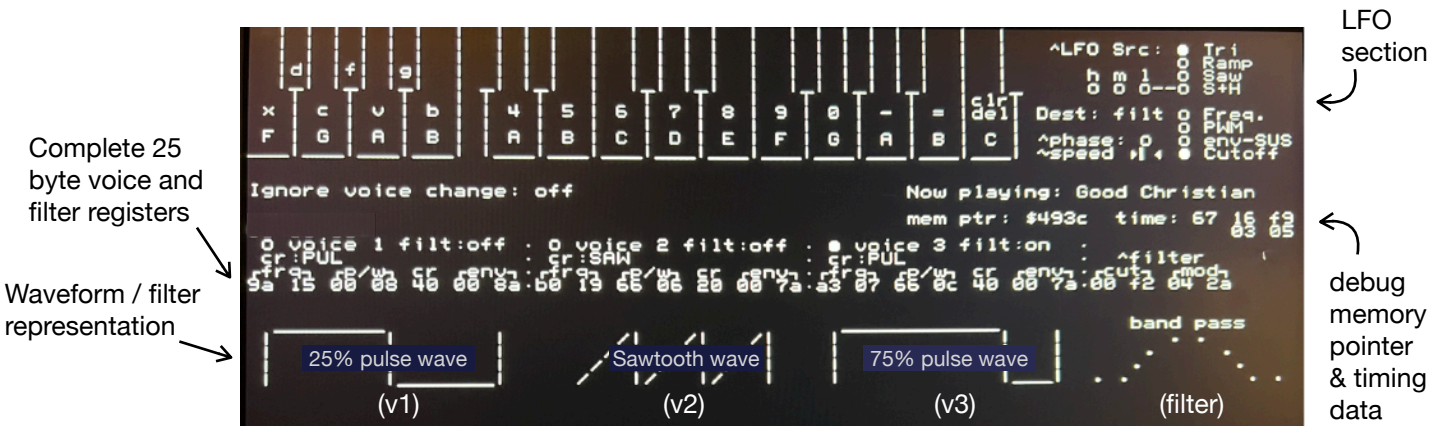
SIDlab is distributed as a PGX file which may have been accompanied by this document within a .zip file. The .zip file also contains Commodore's "Christmas Album" as `.byte` statements, should you wish to examine the byte stream/command structure of the original; useful in your own on-platform projects and also, for processing with modern languages such as Python and NodeJS.

To execute, type the following from SuperBASIC or the equivalent from MicroKernel DOS:

```
/- sidlab_b2.pgx <enter>
```

At startup, music will begin. The instructions and on-screen text is similar to the FoeniXmas demo distributed on December 24th 2023, however, absent the high resolution Christmas Tree graphic. Pressing the space bar will pause the player and alter the display; you will notice several differences:

a) tagging of the hex notation depicting the current SID data phrase with bit decoding; b) rudimentary 'graphic' representations of any of 23 waveform and filter settings; a keyboard guide, and other settings and controls including a software based multi-function low frequency oscillator (LFO) section.



Screenshot from an early beta 2 build

Limitations

- On Jr. systems, there is no reliable way to sense the presence of a 2nd SID (or even a single SID), however, output mode can be switched from *stereo* to *dual mono*. See pg. 9 for more on this topic.
- There is limited stereo field panning and CODEC settings are dependent on system defaults and on the Jr., jumper settings. Otherwise, any CODEC config setup by MicroKernel or SuperBASIC will stand.
- For best results, F256K users may opt to leverage a PS/2 keyboard plugged into the mouse port. The keyboard matrix of the built in keyboard (and Commodore 20-pin connected keyboards on the Jr.) prevent some key combinations from registering correctly. (more on this topic below)

Planned Features

- Mouse support and a graphically oriented interface - for easier selecting and modifying parameters
- More flexibility in assignment of 6 voices and full Wolfson WM8776 CODEC support
- DIN MIDI 'in' support (play your SID with a MIDI keyboard); MIDI 'out' (play the loaded SID track on your attached MIDI keyboard). NOTE: this will require additional hardware, currently under test.
- Song load for a population of COMPUTE! Sidplayer formatted files
- A secondary full-featured LFO and at least one software controlled looping envelope
- Patch save; to write settings of a given voice to disk such that it may be loaded and played by your own SuperBASIC program. Friend of Foenix Rising, Ernesto Contreras, has signed up to help. Look forward to a Foenix Rising article on the subject later this year!
- Options to enable a PSG based percussion (noise) track or a metronome, useful in modeling the timing of envelopes or an LFO

Interface - a bit of old school fun

SIDlab's interface is intentionally low-budget and does not require a joystick or a mouse (deference to Jr. owners who may not have these ports wired). Future versions will strive for mouse and keyboard parity.

Visually, you'll note that there are few frills; in fact, SIDlab uses very few of the F256's special characters and none of the graphic features. The justification for this design decision may appear unusual.

As a youth, many hours were spent on Brookhaven National Lab's VAX PDP 11 connected @ 300 baud. Their system ran an early version of Bell Labs Unix. `Curses` based apps supported cursor positioning on standard terminals (such as the VT100) using nothing but the lower 127 characters of the ASCII set. The slow screen paint was charming and seemed intentional, and a generation of youngsters were introduced to early games such as hangman and rogue, in addition to full screen text editors which did not exist prior.

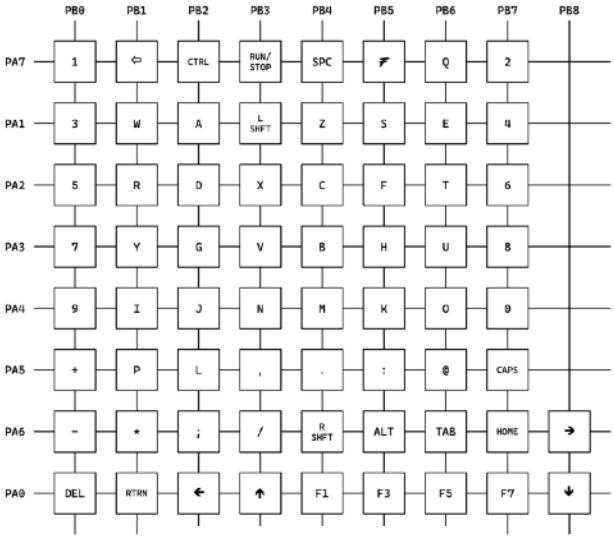
In the *wayback* times, pipes and underscores (underbars) ruled the day. That's what you'll find in this version of SIDlab. A future version will have a more modern interface, but this time around, investing in functionality over flash was the right approach.

The maps on the following pages outline supported keyboard commands in the beta 2 release. Remember, these commands are only available after pausing the player by pressing the space bar.

Keyboard input challenges

Take a look at page 5 and you'll probably agree that the keyboard is not an ideal interface considering the amount of features in SIDlab; but there is a bigger problem. Switched matrix keyboards (which every old-school computer had) leverage cross-bar schemes to reduce the number of sense lines needed from the interface controller; the WDC 65C22. This was (and still is) a precious commodity, not because of the cost of the part necessarily, but because of real estate, PCB lines, and the software overhead to manage it.

If you've been following the development of the Commander X16 project, you probably caught that they omitted the 2nd 65C22 (for the user port) and now bundle it with additional RAM as a 'for cost' option. Savings of $5 or $10 per part across 1,000 units equates to measurable savings when profit margins are slim.



F256K keyboard matrix

On the F256K, we live with this legacy and for the most part, it goes unnoticed. Except SIDlab uses nearly every key on the keyboard and thus, certain odd combinations cause this conflict to manifest.

Inside the kernel, port scanning occurs at lightning-fast rates and debounce and interrupt servicing occurs constantly. The likelihood of a collision during normal typing is not high, in fact, it's extremely low. Engineers made sure that human typing works! Playing music on the other hand, (holding a non-meta key while pressing a second key) is much more likely to be problematic depending on which row or column is being scanned at any particular moment in time.

For F256K users (or the rare F256 Jr. user leveraging the 20-pin Commodore keyboard), this means that an 8 x 8 (or even the 8 x 9) matrix will not be able to reliably sense combinations such as 'x' and '5'. This has not yet been tested with the latest FPGA and kernel, but an update to this doc will be produced shortly.

What to do? Determine which combination does work and avoid those that do not. Or ideally, plug a PS/2 keyboard into your F256K's mouse port. That works too. The hope is, MIDI will soon be viable.

A quick description of 'scope' and how to navigate dozens of key commands

With so much squeezed into a small keyboard, it's easy to lose track of how keys tie to functions. A good way to sort this out is to think about layers by scope: either "voice" scope or "global" scope.

While there are exceptions, *Voice scope* only affects the selected voice (voice *radio buttons* appear once the soundtrack is paused). *Global scope* commands, generally alter settings that apply to all voices or perform utility functions. An example of the former is to choose the noise waveform with voice 1 selected; an example of the latter is to configure the shared filter to 'low-pass' or to adjust the global volume to 12. You'll see that some commands are valid with `ctrl` or `ctrl/alt`. These green parameters are range valid; versus binary toggle vs. single select. Pressing `ctrl` will reduce the value / `alt` will increase the value.

Keyboard map summary (pgs. 5-8):

pg. 5 - Full map: includes every command and control noted with colors, callouts, and arrows pointing every which way. It's not as complex as AVID Pro Tools, but it appears to be getting there.

pg. 6 - Voice scope map: with just a single exception (pulse width adjust), this map is filled with black-circle commands "◯" along with implementation notes for features deserving additional clarity.

pg. 7 - Global scope synth feature map: pertains to shared elements such as the filter, the global volume control, and the LFO. Commands to play all 3 voices in unison or to quiet all voices appear here as well.

pg. 8 - Performance and System features: This map is the 'everything else' view and you'll find a tiny piano keyboard represented here (C-major right hand and a few sharps/flats on the left hand, if you please).

Two other quick things, a simple sequencer/player just for fun

For years, synth enthusiasts have been creating YouTube videos on analog synths, using sequencers to loop "**On The Run**", the famous Pink Floyd sequence from *Dark Side of the Moon*. Now you can too! Pressing '`;`' starts an 8-note sequence at a suitable bpm. The rest is up to you. Alter the envelope, attach the LFO, and change filter settings, then play along with some low or high notes on voices 1 and 2. As an added bonus, you'll notice a hi-hat tracking in stereo in the background. This is not from you SID chips, it's generated by the dual FPGA based PSG instances. Of course, you'll need an additional pair of LFOs, and full control of a 2nd SID (primarily for the 2nd filter) to be convincing, but this will get you started.

In addition to this, pressing '`'`' kicks off a sequence from Rush's "**Vital Signs**" track from 1981's *Moving Pictures* album. The original song repeats an 8-step doubled sequence, but this rendition adds 8 more steps modeled after Geddy Lee's frenetic bass line, stretched to 64 steps with a simple turn-around. The hidden trick in this track is the echo which is played on a 2nd SID (if present). There is yet another way to produce an echo within SID-lab, but you'll need to catch up with episode #8 of the *12DoC* YouTube series for that.

Hopes and dreams for SIDlab

Aside from addressing limitations and adding features, I hope SIDlab gives you something new to do with your Foenix F256 platform. Millions of SID equipped machines have been playing other people's music and sound effects for ages; now you can craft your own. Sure, it's fun throwing chip-tune style data at the SID with players and trackers, but in my opinion, the legacy of the 6581 deserves more. In the early 2000's, a product from Swedish synth maker Elektron called SIDSTATION came close and had a big following, but it was expensive and only supported a single SID. The F256 and SIDlab is the perfect combo to do more, especially once graphic features are integrated.

Audio synthesis (especially within integrated ICs) has a storied past that parallels the development of microcomputers in more ways than you might imagine. Foenix Rising will have more on this subject when the project picks up again, later in the year. Until then, I'll be taking on bug fixes and minor enhancements.

SIDlab
beta 2

Legend: blue - global scope (requires ctrl + a key)
green - shift value up / down with alt (+) or ctrl (-)
black - voice scope (pertains to selected voice)
- A D SR envelopes bound to H U J K keys
- LFO rate controls are unshifted

[ctrl] **quiet** all 3 voices via gate 'off' (begins envelope release cycle) or for voice, disables (sets to 'OFF')

[ctrl] play voice ***n*** via gate 'on' (until silenced) voice (**1-3**)
radio button will move to the selected voice (default is '1')

"On The Run" 8-step sequence
Pink Floyd - *Dark Side of the Moon*
E G A G D C D E

"Vital Signs" 64-step sequence
Rush - *Moving Pictures*
D A F A B♭ A C B♭ - D A F A B♭ A G F

[ctrl] **ignore voice changes** (forces all but frequency edits to stick, despite tracking)

enable filter routing for voice or **echo**
toggle the sequencer (when in stereo)

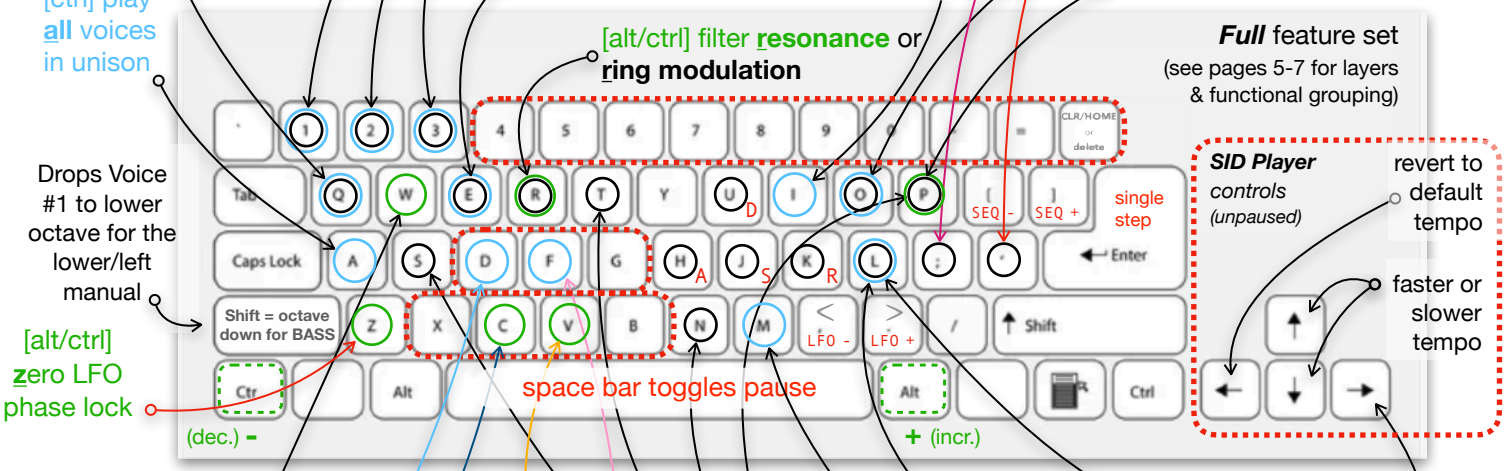
oscillator sync or **output** mode

phase lock (LFO to gate of a voice)

[ctrl] play **all** voices in unison

[alt/ctrl] filter **resonance** or **ring modulation**

Full feature set
(see pages 5-7 for layers & functional grouping)



SID Player *controls (unpaused)*
revert to default tempo
faster or slower tempo

Drops Voice #1 to lower octave for the lower/left manual

[alt/ctrl] **zero** LFO phase lock

[alt/ctrl] coarse pulse **width**

[ctrl] **dump** all SID settings

[alt/ctrl] filter **cutoff**

[alt/ctrl] **volume**

[ctrl] advance **filter** mode (aka type)

[ctrl] **mute** all (forces volume 0; gate & envelopes are not altered)

waveforms:
- **saw**tooth
- **tri**angle
- **noi**se
- **pul**se

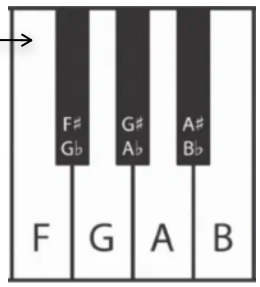
LFO dest.
To pitch, pulse width, envelope SUStain level or filter cutoff

[ctrl] cycle **LFO** type (triangle, ramp, sawtooth, sample & hold low-med-high, or stopped)

skip to next song
(If applicable)

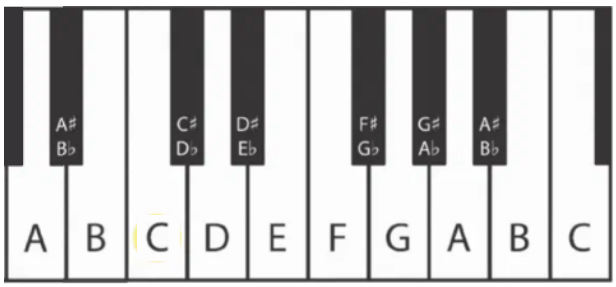
Voice 1
(F2 - B2 incl. sharps)

these 7 keys will drop to the 1st octave when shifted !!



Shift = octave down for BASS

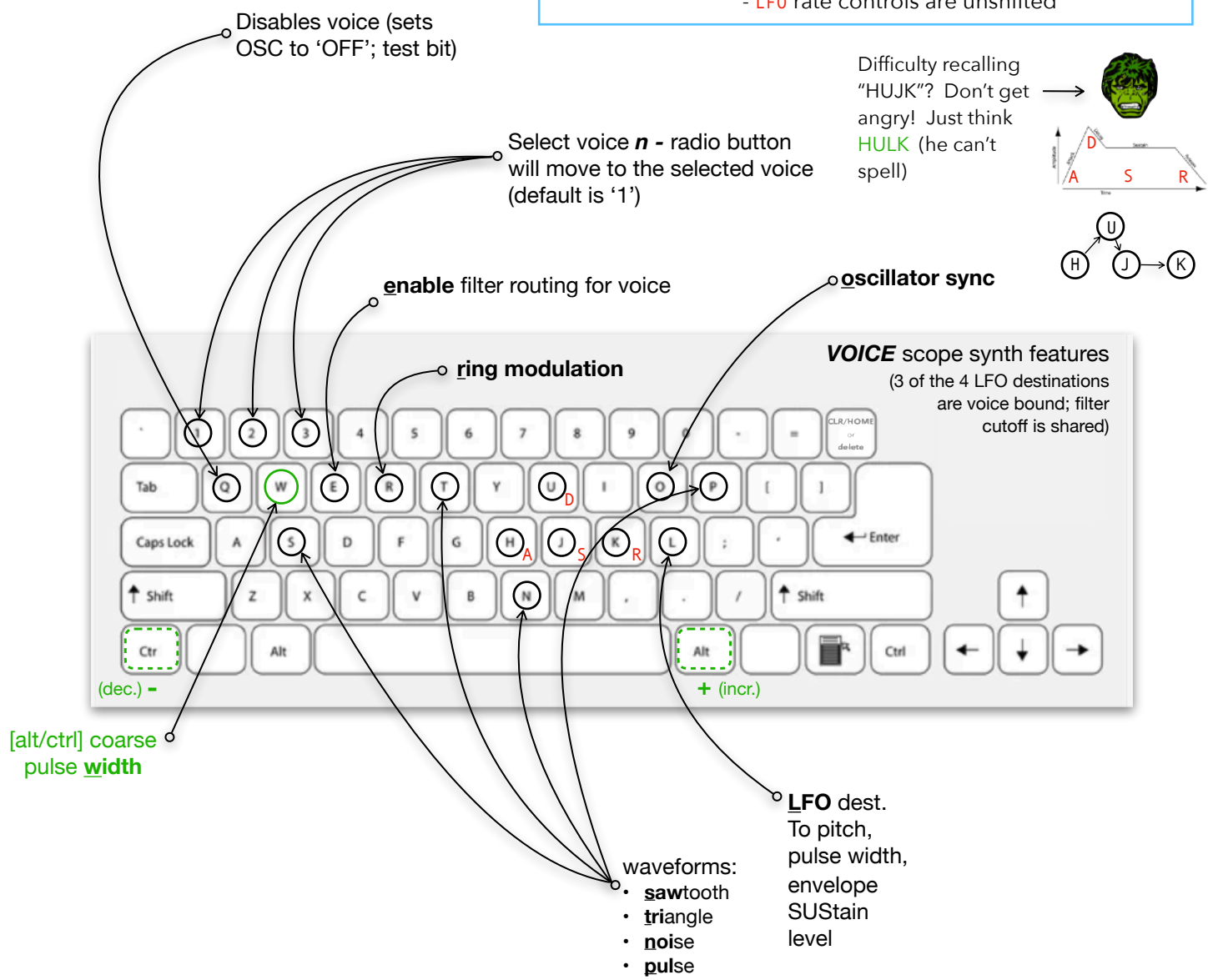
Voice 2
A3, B3, C4-B4, C5 (sharp/flats are not supported by this keyboard)



Line draw colors are arbitrary, but intended to provide clarity on this crowded diagram. Did it work?!

SIDlab beta 2

Legend: blue - global scope (requires ctrl + a key)
green - shift value up / down with alt (+) or ctrl (-)
black - voice scope (pertains to selected voice)
- A D S R envelopes bound to H U J K keys
- LFO rate controls are unshifted



Notes:

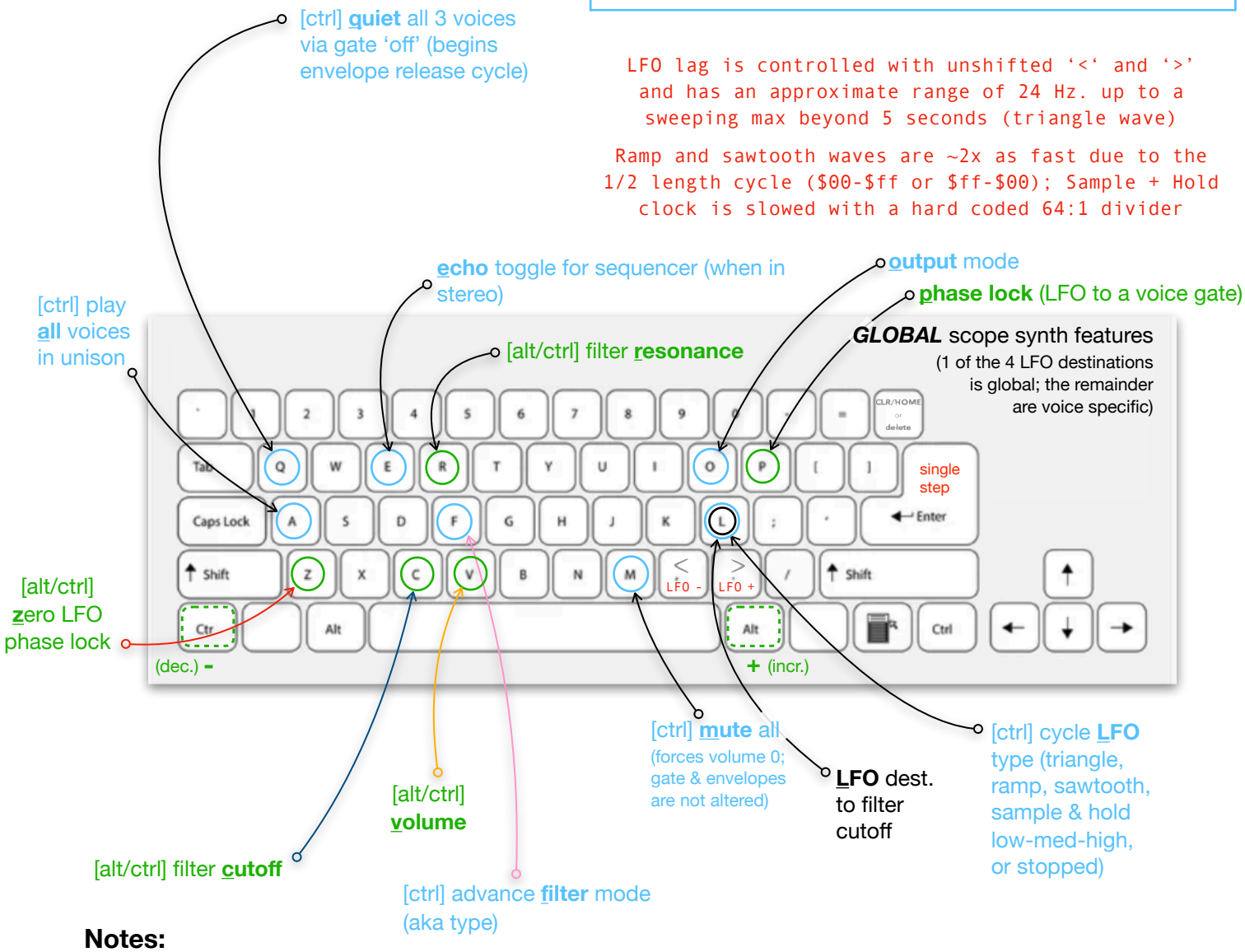
- *Unshifted* controls at this layer affect SID settings for a single voice (either 1, 2, or 3) as focused by numbers with the black circle above.
- In total, there are 7 bytes worth of parameters, representing 21 of the 25 'writable' values in the SID chip.
- There is one exception to the 'unshifted' nature of these controls:
  - Pulse width, when selected for a particular voice, can be altered with `alt` (shifts the width to the right) or `ctrl` (shifts width to the left) along with the '**w**' key; represented by the green circle and callout above.
- Note the red ADSR notation, bound to '**h**', '**u**', '**j**', '**k**' in the center of the keyboard. Pressing any of these keys will bump the value of the 0 .. 15 nibble for the respective envelope stage of the selected voice (wraps at 15).
- Beta 2 of SIDlab features a free-running but voice lockable, mutli-waveform LFO (discussed further below). While global in nature, it can be bound to any of the 3 voice destinations noted above, associated during assign. It can be assigned to a 4th destination (filter cutoff), discussed on page 7. Phase lock is also discussed on the next page.
- It should be noted that ring modulation and oscillator sync depends upon (and will affect) a round robin relationship of OSC pairing. Please consult the official SID spec sheet and experiment with these complex and powerful chip features.

SIDlab beta 2

Legend:		
	blue	global scope (requires ctrl + a key)
	green	shift value up/down with alt (+) or ctrl (-)
	black	voice scope (pertains to selected voice)
		- ADSR envelopes bound to H U J K keys
		- LFO rate controls are unshifted

LFO lag is controlled with unshifted '<' and '>' and has an approximate range of 24 Hz. up to a sweeping max beyond 5 seconds (triangle wave)

Ramp and sawtooth waves are ~2x as fast due to the 1/2 length cycle ($00-$ff or $ff-$00); Sample + Hold clock is slowed with a hard coded 64:1 divider



Notes:

- A note about `ctrl` and `alt` functions; on systems with PS/2 keyboard (either a Jr., or an F256K with a PS/2 keyboard attached), either `ctrl` or `alt` key will work equally; most have two of each. The F256K, however, only has a left-`ctrl` and a right-`alt`, hence the green outlining above.
- `ctrl`-'**m**' mutes the system via master volume, zeroing (low nibble of $D418 / $D518). `alt` / `ctrl`-'**v**' is required to adjust the volume following a mute. Envelope generators with long delays will eventually (up to ~8 seconds) time out; voices for which no gate-off has been triggered will continue to drone, despite volume cut.
- `ctrl`-'**p**' for phase-lock can be used to force a restart of the LFO period for voice gates 1 or 2. This will also work for voice 3, but since there are no *play* keys for this voice, the sequencer gate is attached. Otherwise, the LFO is free-running. An important distinction between rate-sync (not supported) and phase-lock is the latter restarts the period at each gate-on event. The result is greater control over the timbre of a voice, verses locking gate-on and a surgical full-phase completion (sometimes desirable when binding to a MIDI clock).
- Take note, there are several filter configuration pitfalls to be aware of: (a) When a mode *is* selected but no voices are enabled, there will not be any discernible change to the resulting output. (b) The inverse of this is problematic; If a voice is enabled to route through the filter but no filter mode is selected, audio for enabled voices will be blocked. (c) Depending on voice frequency and filter cutoff, some tones can be difficult to hear or completely absent from the resulting audio output; if in doubt, choose a different filter mode, alter the cutoff, or attach the LFO. (d) Boosting the resonance is handy in provoke a less than enthusiastic filter into action!

SIDlab
beta 2

Legend: blue - global scope (requires ctrl + a key)
green - shift value up / down with alt (+) or ctrl (-)
black - voice scope (pertains to selected voice)
- A D S R envelopes bound to H U J K keys
- LFO rate controls are unshifted

[ctrl] **quiet** all 3 voices via gate 'off' (begins envelope release cycle)

[ctrl] play voice ***n*** via gate 'on' (until silenced) voice (**1-3**)

[ctrl] **ignore voice changes** (forces all but frequency edits to stick, despite tracking)

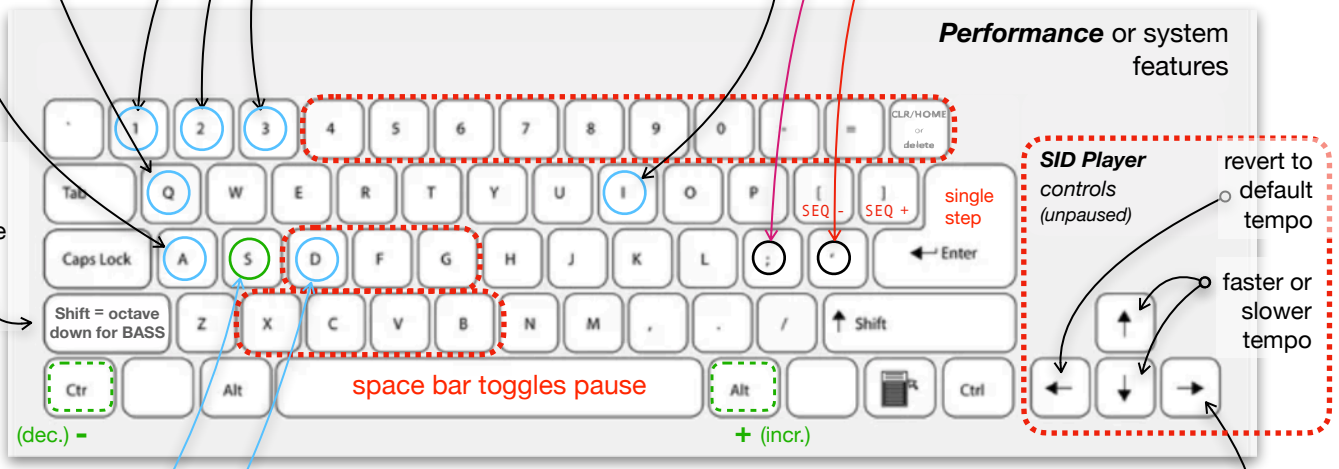
"On The Run" 8-step sequence
(~165 BPM) Pink Floyd - *Dark Side of the Moon*
E G A G D C D E

"Vital Signs" 64-step sequence
(~143 BPM) Rush - *Moving Pictures*
D A F A B♭ A C B♭ - D A F A B♭ A G F

Sequencer lag can be adjusted from the default with '[' and ']'

[ctrl] play **all** voices in unison

Drops Voice #1 to lower octave for the lower/left manual



[alt/ctrl] **sequencer** rate

[ctrl] **dump** all SID settings

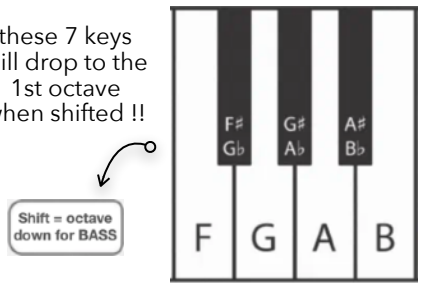
Notes:

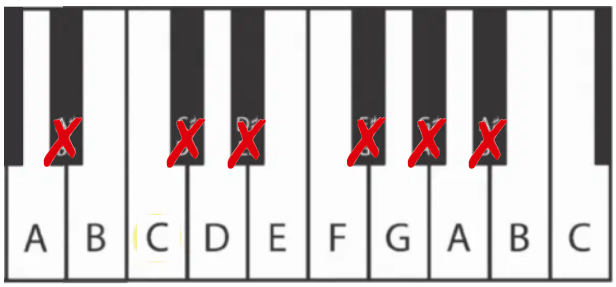
- Until disk routines are developed, `ctrl`-'**d**' can be useful to take a snapshot of SID settings for later use in your own programs
- Pressing **ENTER** (or joystick button) will single-step a paused player
- `ctrl`-'**a**' actuates gate-on for all 3 voices at current settings, but the release stage will not commence until a gate-off (note played or `ctrl`-'**q**')
- `ctrl`-'**i**' will toggle ignore voice change mode; causing the player to ignore all timbre changes except frequency. If 'off' (indicated on-screen), only voice quiet will be obeyed; quieting is useful for 'soloing' another track to observe changes in timbre from the main player.

Voice 1
(F2 - B2 incl. sharps)

Voice 2
A3, B3, C4-B4, C5 (sharp/flats are not supported by this keyboard)

these 7 keys will drop to the 1st octave when shifted !!





Caveats, quirks, work in progress, and hidden talents

As a beta release, SIDlab is bound to harbor quirky behavior and pesky bugs. There will be issues, and your help identifying and reproducing them will result in a better product. Here is a partial list of known issues & opportunities and some additional feature documentation which was not fully discussed on the pages above:

- Timing imperfections - SIDlab beta 2 includes an imperfect time-slicing scheme, currently not leveraging interrupts or kernel services other than for scanning the keyboard and joystick. Managing the LFO, sequencer timing (gate on/gate-off), stereo echo, background tasks, and input concurrently is tricky and while this version does a decent job, it is not cycle-accurate and there are some circumstances where latency will arise. You'll notice this as the sequencer clock hits its minimum lag between steps. In such cases, the amount of cycles spent on overhead is out of balance with the 'worker' portion of the thread. You'll also notice sluggishness when holding down a metakey such as control (doing so causes spurious events to backup the event queue, causing the key-condition tree to waste cycles). A future version of SIDlab will leverage interrupts directly, and possibly a custom keyboard handler. This may become necessary if/when MIDI is introduced (the MIDI clock is chatty and synching LFOs and notes to an external drum and arpeggiator source is time critical and obvious when off).
- Since it was developed quickly, key handling in general is sub-optimal. A long string of conditionals (~70) test for various combinations of valid commands and some classes of control ought to have priority over others. Due to this, SIDlab is not yet leveraging a system to reject invalid keys, in fact, many of the non-F256K keys will trigger ghost functions (e.g. pressing a cursor down on a PS/2 keyboard) sends the same *cooked* code as `alt`-"**v**". This will be optimized as a decision is made about whether to build a kernel better equipped for real-time.
- There is no modulation *amount* yet; think of a MOD wheel on a synthesizer applying a variable amount of modulation to a destination. To compensate, the upper and lower values of modulation is artificially limited in some cases. One example of this is to limit frequency modulation to a small set of values. The result is good for vibrato, but not for UFO sounds. Another example led to creating tailored ranges for sample + hold which now has 3 options (low, medium, or high range; roughly equal to top, middle, and bottom 3rds). This will ultimately be bound to an external continuous controller (CC1 is customary) assuming MIDI can be integrated. In a non-MIDI use case, I'm considering an accumulated velocity method tied to a momentary key such as CAPS LOCK.
- Sequencer features & limitations: By default, the "VITAL SIGNS" sequence includes a stereo echo if your platform is so equipped (two SIDs) *and* the output mode is set to "stereo". "ON THE RUN" is not; however, you can toggle this by pressing `ctrl`-"**e**" while the sequencer is running. This will only work for the current session (not if stopped and restarted) and it will not work in the song player. Likewise, the PSG percussion track on "ON THE RUN" is enabled by default and not available for "VITAL SIGNS". There is currently no option to alter this behavior, however a future version of SIDlab will include more percussion and sequencer options.
- If stereo is selected and the SID player is unpaused, you will notice phasing of the pulse wave between left and right voices; this is not true stereo since it's the same data between SID chips, however the natural crossover point of pulse waveforms create the sonic illusion of panning between channels. This only applies to pulse wave.
- If stereo is selected in paused mode, you'll notice that voice 1 is sent to the first SID and voice 2 is 100% to the second. Depending on CODEC and jumpers, this will be left and right channels respectively. A future release of SIDlab will incorporate channel dynamic controls via the built in Wolfson CODEC chip. If you desire both SIDs to output to both L and R channels, switch from "stereo" to "dual mono" using `ctrl`-"**o**".
- There are several on-screen hints for `ctrl` ('^') based commands; this was only able to be added where space allowed; `ctrl`-"**l**" for LFO source/type, `ctrl`-"**p**" for Phase lock, `ctrl`-"**o**" for audio output mode, `ctrl`-"**f**" for filter mode, and `ctrl`-"**i**" for ignore voice change. There is alot to remember, especially when first using SIDlab. This, in conjunction with the layers described above, should help.
- False echo - aside from the 'slap-back' echo included with the sequencer, it is possible to configure a ramp or sawtooth sourced LFO with phase lock, a LFO destination of envelope-SUSTAIN, and a long release to create a trailing echo on notes played sporadically. Try this at different rates. This is discussed in great detail in episode 8 of the 12 DoC SID series under *advanced use*.
- The pulse width LFO destination only affects the lower 8 bits of the register while the `ctrl`/`alt`-"**w**" command only alters the upper 4 bits [$00..$0F], hence the "coarse" description.
- The opposite is true with filter modulation; the destination is bound to the highest order byte (a full 8-bits). Filter Cutoff is an 11-bit value and while `ctrl`/`alt`-"**c**" allows any value across the range to be selected, the LFO will only alter the high-byte and leave the lowest 3-bits unchanged. (see the 6SID spec sheet for more).