SIDONIA

/// A DAW FOR THE C64*





USER MANUAL

Article for INPUT ONE / Magazine for Synthesizer Culture & Tech Obsession by Joan Rue

Commodore's C64 is a chapter of computer history very much its own. Above all because this contraption crossed the shop counter more often than almost any other computer. For that reason there are still plenty of working units around today, in the form of attic finds and specimens that are to be had elsewhere on the market. It's a good thing that this machine, in terms of construction, comes from a different league of manufacturing. Solidly built and fairly easy to service, provided you know what you're doing and have the right tools at hand.

When I first sat in front of a C128 in 1991 and gazed in amazement at the pictures flickering across the monitor, I was done for on the spot. The sound and, above all, the graphics had burned themselves into my own memory like a ghost image on an old CRT monitor. We are, after all, precisely that generation which witnessed the beginnings of computer history right up to today's internet, awash with advertising. Today's computers naturally have more processing power than this lump, in which the keyboard and all its functions share one and the same shell. With not even 1 MHz of power, what comes out of this milestone, affectionately known as the breadbox, is truly more than remarkable.

With regard to memory usage and the aforementioned performance, this box is really remarkably efficient. Naturally, a program on this computer has to be well structured, otherwise the general graphics and the entire user interface of any prospective program behave like a flip book in slow motion. Back when I sat in front of that C128 D looking at the graphics, I would never have believed that some of these programs are so complex that they almost amount to a Ninth Symphony. Probably only very few people knew about that at the time, and I didn't have the faintest clue.

The music from this casing was extraordinary, and 15 years ago I made up my mind to coax a few tracks out of the legendary SID chip. Even back then, the search for something suitable proved difficult. I knew Fasttracker 2 from the Amiga, and there were a few trackers on this old platform. But working with those programs was peculiar. So I wrote everything in pure code, which unfortunately produced no particularly satisfying result beyond simple melodic sequences. When I turned my attention back to this machine, it was difficult once again, since the trackers are evidently a bit inaccessible. Especially if you have never seen hex code before. It gets easier with time, and the encoding

makes sense insofar as it is extremely simple and, in terms of display, very memory-friendly. Above all because the computer runs in text mode this way. The normal character set can be loaded faster and processed correspondingly fast.

One day I was sitting in front of it again and had truly had enough. And I thought to myself that it ought to be possible to pursue a different approach, one that could be visually more understandable on the one hand and more efficient in its way of working on the other, for all those people who simply don't want to dig through hex code. The decision was made, and it pushed me to a great many limits. Sleep is overrated? Well, that really isn't correct. Because this machine has its secrets and a slightly stubborn nature when it comes to the individual registers. So it's no burden to sit in front of this magic box for so long. However, at the start I would never have suspected that it would really be this long. And I never strayed from my plan to solve the whole thing more graphically, but I completely discarded the graphics mode right at the beginning, since the processing power required purely for the editor itself would already have been too much. It would merely have been a program stuttering along like a dying tractor. Text mode, in the end, brought a new glimmer of hope to the whole development. Yes, graphically the user interface sits closer to the year 1978 than to 2026, but it doesn't always have to be pretty, as long as it serves its purpose. I first had to make my peace with that thought too, but bit by bit I made the best of it and crept, piece by piece, around the performance limits. Just like a marten on its nocturnal prowls, which does after all find its way into the engine bay and sinks its teeth into the wiring harness.

The name of the project is "SIDONIA". I had in mind the region on Mars that is home to the famous Face on Mars. A seemingly unreachable mystery, and yet it is visible. Well, the lettering in the title still dates from the early phase, and since I had to watch the memory, it stayed that way. After almost 6 months I was finished. It works and is a DAW for the C64, created along the lines of a somewhat more modern way of working. So that more people gain access to the possibility of coaxing entire tracks out of the legendary SID chip. Whether for pure sound design for music, when it comes to games, or simply artifact sounds and samples. A great deal should be possible with it. And the SID itself certainly has its clearly stripped-down limits, but I have tried to fit in pretty much every function this little silicon beetle has. Until, among other things, I ran out of room. On the screen on the one hand, and in memory on top of that. The demo song was made on this DAW, along with a few test loops that were lost to crashes. The final phase was, once again, special, but now there is the option of using the user interface with a cursor, and on top of that the program has support for a C64 1351 mouse, which can be connected via the joystick port. When this feature worked for the first time, the mouse pointer merely danced wildly across the screen, because I had got the address wrong. The control routine was simply capturing the noise of the SID chip and rendering it as a control value.

/// THE MOUSE POINTER THAT DANCED THE NOISE

I want to linger on this anecdote a moment longer, because it says a lot about the nature of this machine. The 1351 mouse doesn't report its movement digitally, but via the two POT lines of the SID ... the same analog inputs that paddles otherwise hang on. The SID samples these lines only about every 512 clock cycles. And here lies the catch: the KERNAL constantly scans the keyboard and in doing so keeps writing to the register $DC00. But that very register also switches the analog multiplexer that determines which port is connected to the SID. If you now read the POT values directly after switching, you still get the old value ... the one from the keyboard or from port 2. Under heavy load, by chance enough time passed between switching and reading for the SID to deliver a fresh value. The pointer moved. But as soon as no animation was running and the main loop was racing through unhindered, there was no time left between switching and reading ... the SID hadn't sampled anything new yet, and what I read out was not the hoped-for mouse movement but simply the chip's background noise. (Really great) The pointer danced across the screen as if it had a will of its own. (Impractical) The solution was unspectacular and instructive at the same time: select port 1, then wait at least two full SID sample periods ... over a thousand cycles ..., always discard the first reading and only use the second. Since then the value has been stable, no matter how fast the loop runs. It is one of those places where the C64 teaches us that you mustn't work against its timing, but with it.

And since this wretched little pointer accompanied me through the whole development anyway, let me tell a second chapter of its story right away. In the final phase I switched it from a character written into the picture to a real hardware sprite. That sounds like a minor thing, but it is the difference between a pointer that paints the characters beneath it black and one that hovers cleanly and translucently above the interface. A sprite reads its shape continuously from memory … and exactly that became a trap once again: the routine that clears the screen also overwrote the sprite pointer every single time, so that the poor arrow suddenly drew its shape from the middle of the song data block and appeared as a garbled pattern that nevertheless dutifully moved with the mouse. (= Can't be bothered anymore!) It is astonishing how often the same lesson comes back in new clothing: on the C64 you have to know who writes to which memory and when. (CATASTROPHE!)

And there is a third chapter too, because the mouse had one last surprise up its sleeve. For weeks, while I was working, the program would now and then jump into another window for no reason. No crash, no pattern, just: click, and suddenly I was in the mixer rack. For a long time I blamed it on the mouse movement, on glitches in the POT values, on the timing of the click. The truth was simpler and older than the program. The left button of the 1351 hangs on the same line as the fourth row of the keyboard matrix. The KERNAL polls the keyboard every frame, column by column, and if my click fell exactly into the polling of the first column, the KERNAL saw a pressed key there. Row four, column one, that's F1. And F1 calls up the rack. In other columns the click turned into a letter or the space bar. A joystick in port 1 "types", people knew that back in 1982, and I got to learn it all over again in the year 2026. Since then SIDONIA discards every key that coincides with a pressed mouse button. Twelve bytes, and the haunting was over.

A fourth chapter only came to light on the real hardware, and it was the most instructive. In the emulator everything ran cleanly; on the breadbox the step sequencer suddenly drew streaks through the piano roll, a row of little "e"s changed colour beneath the grid, and switching to the song window ended in a red block. It looked as if one part of the program were writing into another. That is exactly what it was, except that the writer wasn't SIDONIA but the KERNAL. My pointers lived on the zero page at $F2 to $F9, an area the documentation lists as free as long as no BASIC is running. What it doesn't say: the KERNAL's keyboard scan, which runs along after every interrupt, deposits its pointer to the decoding table at $F5 and $F6 on every key press. So if you play a note on the keyboard while the sequencer is running, two of my pointers suddenly point right into the middle of the ROM, and the next cell that is supposed to be drawn lands somewhere, just not on the screen. In the emulator this went unnoticed because my test setup never really let the keyboard scan run alongside. Since then all pointers live at addresses the KERNAL is guaranteed to leave alone, and the simulator I use to check the program replicates the KERNAL's complete interrupt tail, keyboard scan included. The C64 simply has no operating system that forbids you anything. It has one that allows you everything, including the mistake.

/// TWO WAYS OF OPERATING — MOUSE AND CURSOR, CLEARLY SEPARATED

SIDONIA is divided into four working windows that map the path from the single sound to the finished track. Anyone who has ever worked in a modern DAW will recognise the logic immediately: first the channel, then the sound, then the melody, then the arrangement. So you wander from window to window through your own production.

Everything is operated via two kinds of input that deliberately keep out of each other's way. Having a mouse pointer and a keyboard cursor side by side at the same time quickly leads to confusion on the C64 … double pointers, unclear focus. That's why I opted for a clear, unambiguous separation instead of letting the two run in parallel all the time. The switch happens quite naturally out of the flow of operation. If you press a cursor key, SIDONIA switches to keyboard mode: the mouse arrow disappears, and a block cursor takes over, which you then move cell by cell with the arrow keys and trigger with RETURN. If, on the other hand, you click the left mouse button, mouse mode returns: the cursor disappears, the mouse arrow reappears, and you point freely at whatever you want to change. So exactly one input method is always active, and you decide for yourself which … with the arrow key in one direction, with the click in the other. I wanted to solve it differently, which is why I'm saying this so plainly here, but I had already cobbled it together another way beforehand and then simply left it that way.

And this has a really pleasant side effect, entirely in the spirit of this machine:

in keyboard mode, the laborious polling of the 1351 mouse pauses completely. That POT reading, which spends over a thousand clock cycles per frame in pure waiting, is dropped the moment you're working with the keys anyway. So the separation is not just a matter of clarity, but also noticeably spares the little processor some work. Besides keyboard and mouse, SIDONIA also understands the joystick in port 2 ... the fire button triggers there just like RETURN and the left mouse button. So there are no cryptic keyboard shortcuts you have to learn by heart; you point at what you want to change and confirm.

A word about clicking itself, because that too changed in the final phase. Previously the program only polled the mouse button when it had nothing to draw at that moment. A short click that fell into a screen refresh was simply lost, and the operation felt stubborn. Today the interrupt routine remembers every button press, fifty times per second, and the main loop picks it up as soon as it has time. In addition, a click in the piano roll now redraws only the one affected row instead of the whole screen, and in the arrangement only the one track. Those are the two things that make the difference between "it responds sometimes" and "it responds".

/// THE KEYS THAT APPLY EVERYWHERE

A few keys work throughout the whole program, no matter which window you happen to be in. They are the backbone of the operation, and it's worth having them in your head once.

THE FOUR WINDOWS: F1 calls up the RCK window (mixer rack), F3 the EDT window (sound editor), F5 the ROL window (piano roll) and F7 the SNG window (arrangement). In the SNG window you can additionally reach the same four windows via the number keys 1 to 4, since the F keys are not available everywhere.

THE FOUR CHANNELS: F2, F4, F6 and F8 select the active channel (voice 1 to 4). Everything you edit afterwards refers to this channel.

PLAYBACK: the space bar starts and stops playback. The keyboard is laid out like a classic tracker: the lower row z s x d c v g b h n j m plays one octave, the upper row q 2 w 3 e r 5 t 6 y 7 u the next. This lets you play notes in directly for preview and hear, at any time, exactly the channel you are currently working on. (I also wanted to add a swap for the keys Y and Z, since depending on the device and keyboard layout these can be swapped in the emulator. For reasons of space I discarded that, though. So it may be that two notes on the keyboard are swapped accordingly. So please don't be surprised.)

The header and footer are the same in every window. In the header you'll find the BPM display with a [-] and [+] for adjusting the tempo, as well as the tabs of the four windows. A short click changes the tempo by one; if you hold the mouse button (or RETURN in cursor mode), after a brief moment the number keeps running on its own until you let go. The footer carries the global commands: [play], [stop], the switch between [patt] (pattern mode) and [song] (song mode), plus [save], [load], [rst] (reset: clears patterns, arrangement, swaps and automation, keeps presets and tempo, and switches to pattern mode so the step sequencer plays again straight away) and [exp] (export).

/// WINDOW 1 — RCK: THE MIXER RACK

The rack is the starting point. This is where the SID's four voices sit side by side, each with its own preset, its volume and its state. In a sense, this is where you set up your instruments before you write a single note. Each channel offers the following controls:

Choosing a preset. With pre: [-] and [+] you page through the 16 presets that ship with the program. A new preset is applied to the channel at once ... so the sound is ready instantly. And the presets are not untouchable templates but sixteen memory slots: whatever you change on a channel in the editor ends up in that channel's currently selected preset. So you can tweak two similar sounds into shape on two slots and switch between them, and when you save, they travel along. The same goes for the random button [rnd], which overwrites the selected preset right along with it.

RENAMING A PRESET: Click on the preset's name and renaming begins. You type the new name and finish with RETURN, or you simply click somewhere else, which ends the input as well. While renaming is in progress, every key goes into the name, including the space bar and the keyboard piano; so if you are wondering why the keyboard has stopped doing anything, you have usually just clicked on the name.

MUTING:
The [play] field shows that the channel is playing along. One click mutes it: the field changes to [mute] and the channel stays silent until you click it again (or confirm it with the cursor and RETURN) to set it back to [play].



Nothing is played when you click it; for auditioning, use the keyboard piano.

VOLUME AND OCTAVE:

In the VU line below, < and > set the channel volume between 0 and 15. To the right of the preset name you will also see O: followed by a digit from 0 to 7, for example O:3. That is the octave of the instrument on this channel, as set in the editor (EDT) in the "o = octave" line. It shifts everything this channel plays by whole octaves on playback: 3 is the middle, 2 sounds an octave lower, 4 an octave higher. The rack only displays it, so you can see at a glance which register a channel sits in; you cannot change it here, for that you go to the editor. It has nothing to do with the o field in the piano roll, which only moves the visible section of the keyboard, not the sound.

Volume deserves a paragraph of its own, because this is where the SID outfoxed me for a long time. The chip has no per-voice volume. Every note always shoots up to full level, and the only thing you can influence is the level it sinks to afterwards, the sustain. With a short note and a normal decay, the note is already over before that level is even reached, and with a drum, whose sustain is zero anyway, exactly nothing happens. And that is exactly how it sounded: you turned the control, and nothing changed. So SIDONIA takes a detour that you would otherwise have to build by hand in a tracker. At reduced volume the decay is cut as well, so that the envelope drops immediately to the lowered sustain. As a result the note is genuinely quieter and not just shorter in its tail; on a drum the cut acts as volume. At full level the instrument is left untouched. This channel volume is now also saved and travels along into the export.

THE RACK IS DELIBERATELY KEPT PLAIN:

it is meant to be the place where you keep the overview, while the real fine work on the sound happens one door further along, in the editor.

/// WINDOW 2 — EDT:
THE SOUND EDITOR

The editor is the heart of SIDONIA. This is where you shape the individual sound right down into its registers. Each parameter sits on its own line, and each line is changed by clicking on the value scale. The order in the window follows the musical structure of a sound … from the envelope through the waveform to the finer points of modulation:

A ATTACK / D DECAY / S SUSTAIN / R RELEASE — the four values of the envelope (ADSR), each as a slider.



T TRIGGER — the instrument's trigger mode, switchable between ABS (hard retriggering, tight timing) and REL (tied, legato-style playing).

W FORM — the waveform: SAW (sawtooth), PUL (pulse), TRI (triangle) or NOI (noise).

C CUTOFF — the filter cutoff frequency.

O OCTAVE — the octave, from 0 to 7. Zero is a real value here, not a blank: it is the sub-bass range three octaves below the middle, and the presets kick and sublow sit exactly there.

Q RESON-q — the resonance (filter Q), also from 0 to 7; zero simply means no resonance.

H HARD-S — hard synchronisation (hard sync).

R RINGMOD — ring modulation.

A /// ARP — the arpeggiator, with the patterns OFF, MAJ, MIN, OCT and FIF.

P WIDTH — the pulse width (only audible with the pulse waveform = square).

F F-TYPE — the filter type: OFF, L-P (low-pass), H-P (high-pass) or B-P (band-pass).

M PWM-SPD — the speed of the pulse-width modulation.

G GLIDE — gliding between pitches (portamento).

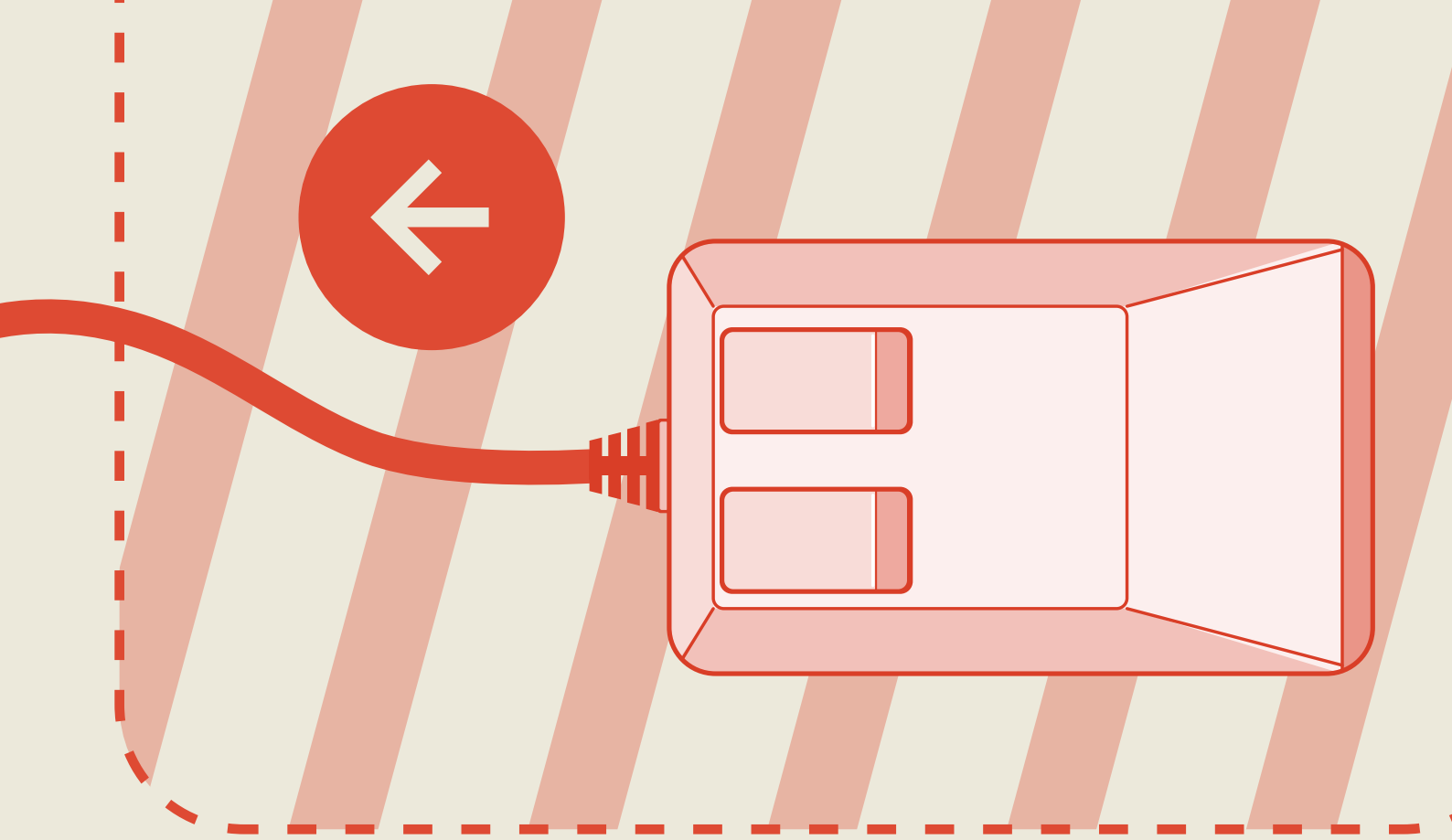
V VIBRATO — the vibrato.

K ATK-WAVE — the attack wave: OFF, NOI, SAW, PUL or TRI. More on that in a moment, because this is the line that breathed life into the drums.

The editor also shows an envelope graph (env-graph) that makes the ADSR curve visible. It is there for illustration and is not clicked.

A word about the trigger switch, because it solves a very concrete problem of the SID. Without hard retriggering an instrument sounds slightly different on every hit, because the envelope does not start cleanly from the beginning … on melodic voices that can be appealing and lively, but on the short SID drums it means that hits are literally swallowed. With the setting ABS (absolute) you force a clean restart of the envelope; REL (relative), on the other hand, lets the notes flow into one another. So you decide per instrument which suits better.

And then the ATTACK WAVE. The demoscene has always had a trick that makes its drums crack and its basses pluck the way they do: in its first milliseconds a note runs through a different waveform than afterwards. A burst of noise before the pulse, a sawtooth before the triangle. In the trackers this lives in waveform tables that you fill in by hand, tick by tick. In SIDONIA it is a single line with five fields. If you choose NOI, say, the SID plays noise for the first three frames, that is 60 milliseconds, and then switches to the instrument's actual waveform. A bass with saw and an ATTACK WAVE



of NOI gets an attack like a plectrum. A kick made of TRI finally gets the thump it was always missing, and a snare made of noise gets a body with PUL in front of it. It feels as though you had cheated, and yet it is only the hardware and the physics. The ATTACK WAVE belongs to the instrument, so it travels along into the presets, into the swaps and into the export.

And while we are on the subject of the attack, a word about what you do not see: the hard restart. The SID has a notorious quirk, the ADSR delay bug. If you strike a note while the envelope of the previous one is still somewhere between heaven and earth, the new attack does not begin at zero but wherever the chip happened to be, and sometimes only a whole envelope cycle later. That then sounds like a click, a plop, or a note that arrives late. Every serious tracker counters this with the same trick, and since VERSION 2.5 SIDONIA does too: one tick before a new note lands on a channel, it takes the gate away and sets attack, decay, sustain and release to zero for that one moment. The envelope drops to the floor in six milliseconds, and the attack starts cleanly from the beginning, the same every time. Legato channels (REL) are left untouched by this, since tying the notes together is the whole point there. You do not hear it as an effect, you only hear that it is gone.

Whatever harshness you still hear after that is the chip itself, and you should know about it before you take it for a bug. The ATTACK WAVE switches the waveform in the middle of the note, which clicks by design, that is what it is for. The SID has exactly one filter for all three voices; if two instruments have different filter types, the filter switches on every hit, and you hear that as a jolt in the sound. If you do not want that, give the filtered instruments the same type. The drum channel borrows the third voice and cuts off a running note there hard, that is just how it is with four channels on three voices. And the MASTER VOLUME, which the automation can drive, is a famous clicker on the 6581 (SID), the very same one the demoscene uses to play its digi samples; if you want to fade out gently, better use VOL per channel. And the noise, finally, is noisy. It is a shift register from 1982, not a plugin.

If you love chance, you will find the button [RND] in the editor's title line. It rolls the dice on every value of the instrument currently being edited, completely from scratch. That is an excellent tool for tearing yourself out of a stuck idea of a sound and stumbling onto ideas you would never have come up with yourself. The SID has a fondness

for beautiful accidents.

A note that will become important later in the arrangement: what you set here on a channel's instrument is binding for the song and gets saved with it. All instruments are stored in the song. If you change an instrument, that change is a fact the track reckons with from then on.

/// WINDOW 3 — ROL: THE PIANO ROLL

The PIANO ROLL is where you write the actual notes. It shows a keyboard along the side and a grid of cells; one click sets a note, another click on the same cell removes it again. The roll always works on the currently selected pattern and channel. In the info line along the top edge you control the framework of the pattern, each via a < and >:

P — the active pattern (0 to 23). Sadly, that was all that would fit. I wanted to solve it with lookup tables, but that would have been too slow, so there are only these 24 PATTERNS.
L — the pattern length, switchable between 4, 8 and 16 steps. You may do this even while playback is running; the sequencer stays in time. In the arrangement, SIDONIA adjusts along with you: if you shorten a pattern, it fills the cells that become free with the same pattern, and if you lengthen it again, it takes exactly those repetitions back out. So the song keeps sounding as it did before and stays the same length, instead of acquiring silent gaps.
C — the channel (1 to 4).
O — the displayed octave, so you can shift the visible pitch range. To the left of the keyboard a green square shows which of the eight octaves you are currently in, and the two fields [+] and [-] next to it page through them as well.
A — the instrument's arpeggiator, switchable on and off right here.

The number of visible steps can additionally be changed with the + and - keys.

LONG NOTES:

A note starts out one step long. It gets long in two ways. Either you click in the gap between two set notes on the same row, in which case SIDONIA joins them into a held bar, and a second click in the same gap splits the bar there again. Or you set a length from 01 to 16 with < and > in the NOTE LENGTH row at the bottom and then place the note with a single click as a finished bar starting from the chosen step. If it sticks out past the right edge, it is cut off there. This was an approach to make up for the lack of moving and copying individual notes, and it has turned out to be the faster tool.

A few rules apply here, which I chose deliberately. A single note on a step where a single note of a different pitch already lies replaces it. Within a HELD BAR, on the other hand, you may set further notes on other rows; the bar stays put. The SID is monophonic per channel, so the bar is taken over by the new note at that point and does not continue afterwards, but that is exactly what sometimes makes a held note more interesting: a dab on top, a variation underneath. Two bars stacked on top of each other are allowed too, as long as they begin on different steps. Only exactly on the starting step of another bar can no note be set, so that nothing gets deleted by accident; click the bar away first, then set the note.

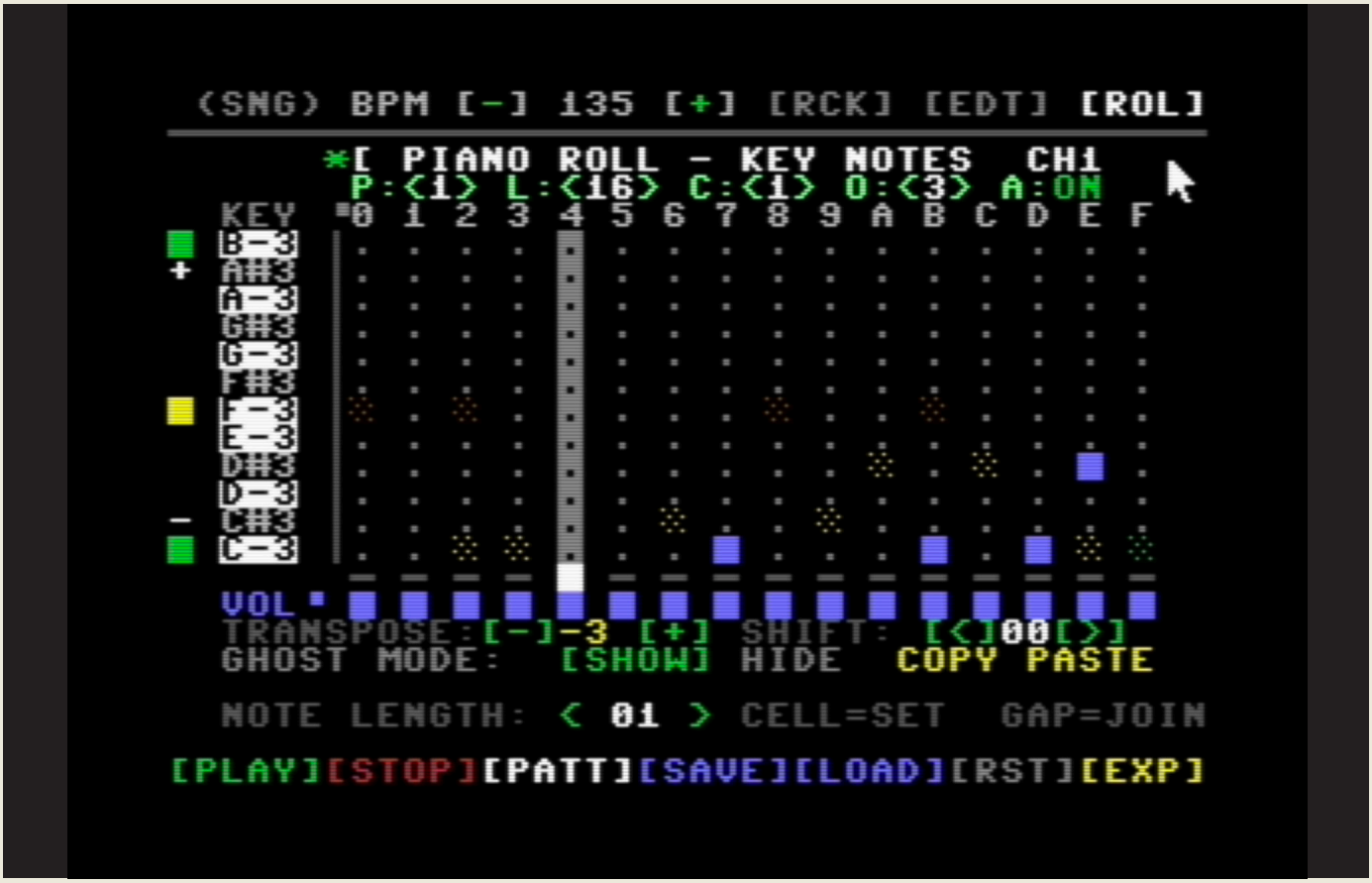
Below the grid lies the VOL ROW with one coloured block per step. That is the velocity of the respective note for the channel shown: LIGHT BLUE full, CYAN loud, GREY quiet, DARK GREY almost off. A click on a block lowers the value by four levels; below zero it jumps back to full. Four clicks, once around. With it you build accents and ghost notes, say a hi-hat that is pulled back on the offbeat. And because the volume is now really audible (see the rack), this row is suddenly worth the effort.

While playback is running, a vertical, inverted BAR travels through the GRID and shows the column the sequencer is currently playing. The column next to the keyboard additionally marks the note currently sounding on this channel, and also whatever you are playing live on the keyboard; the key name itself lights up inverted along with it, so you can see it out of the corner of your eye too. So you see what you hear.

TRANSPOSING AND SHIFTING:

In the row below sit two tools that save me the most time when arranging. TRANSPOSE moves all the notes of the displayed track one semitone down or up with [-] and [+]; the two-digit number in the middle shows how far you are from the original, and turns yellow as soon as it is no longer zero. As soon as even one note would leave the pitch range of eight octaves, nothing happens any more; the track stays put as a whole instead of bending at the edge. SHIFT rotates the track one step to the left or right with [<] and [>], wrapping around within the pattern length, lengths and velocities included. Putting a bassline on the offbeat is thus one click instead of a rebuild. Both counters are kept per pattern and channel.

A special detail is the GHOST MODE:



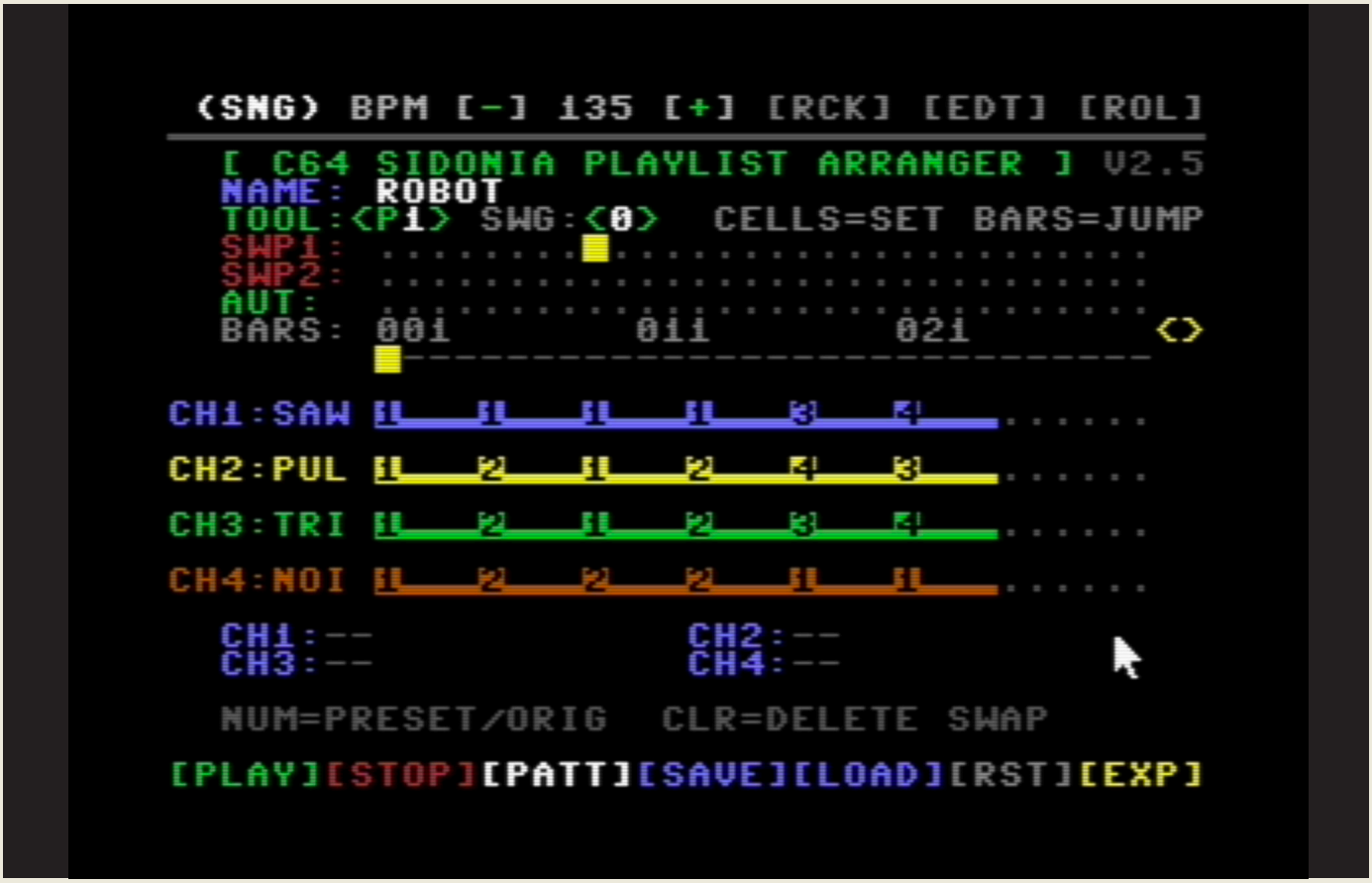
it shows the notes of the other channels as pale "ghosts" in the background, so you can write your melody in the context of the rest of the arrangement. If notes collide with each other, SIDONIA makes that visible by blinking … so you see conflicts beforehand instead of only hearing them on playback. That was an idea of mine for keeping an overview; I hope it is enough. Anyone who finds the blinking gets in the way while writing, because they want exactly that friction, switches it off in the row below: CONFLICTS SHOW/HIDE sits flush beneath the fields of the ghost mode. It is a hint, not a prohibition; two channels on the same note are sometimes exactly what a riff needs.

For working with repetitions there are COPY and PASTE, with which you copy a pattern and insert it again somewhere else, to make faster progress while working. In my tests I noticed right at the start that I was spending too much time setting up arrangements, and so I added this function.

A word about the voices, because here lies a quirk of the SID. The chip has only three real voices, but SIDONIA has four channels. Channel 1 and channel 2 each have their own voice. Channel 3 and channel 4, on the other hand, share the third voice of the SID. So channel 4 is not a hardware voice of its own but a virtual channel that borrows the third voice from channel 3. If both hit a note on the same step, channel 4 wins: it takes the voice, and the note on channel 3 is cut off at that moment. That is not a bug, but the only honest way to make four channels out of three voices. What you put on these two channels is left up to you … the program prescribes no roles.

The shared voice is especially attractive for drums, though. You put the kick on channel 3 and the snare, built from noise, on channel 4. That gives you two percussion instruments on a single voice, without sacrificing one of the two real melody voices for it. And because channel 4 has priority, the snare wins out over the kick when in doubt, just as you know it from a real drum kit, where the hit on the snare covers the decay of the kick. If you deliberately set the two against each other, you get a lively interplay; if you let them overlap on purpose, you create dropouts that can be attractive as rhythmic variation.

In the bundled demo song, by the way, the division is a different one: there, channel 3 holds a bass voice that acts as a kick, and channel 4 a kick that I remodelled into a snare. That simply came about that way while building the demo, because at that point I could not yet rename the voices. It is a good example of how the same shared voice



can be used in completely different ways … sometimes for bass and one percussion sound, sometimes for two pure drum voices. The technique behind it stays the same; only the decision of what to make of it lies with you. A voice can, incidentally, also be used as a bass and for the highs at the same time, by making clever use of the octaves of the piano roll.

/// WINDOW 4 — SNG: THE ARRANGER

The last window joins the individual patterns into the complete track. In SIDONIA it is called the PLAYLIST ARRANGER. Here you work, in practice, with four tracks that run in parallel, and in them you arrange the individual patterns into a song. Each track consists of slots; one click sets a pattern at that spot, a second click on the same cell deletes it again.

The song can be given its own title in the NAME LINE. A click on the name line clears the field and opens the input, so you can start typing freely right away without first having to laboriously clear away the old name. You type the new name and finish the input with RETURN. And should RETURN ever act up on your own keyboard — depending on emulator and keyboard layout, the key does not arrive the same way everywhere — another click on the name line is also enough to leave the input. So you are not stuck.

This is also where you decide whether to polish a single loop in pattern mode ([PATT]) or hear the entire track from start to finish in song mode ([SONG]). When playback reaches the end of the arrangement, it automatically jumps back to the beginning … so the song runs as a loop as long as you do not stop it.

A small but attractive detail on the side: if you switch a pattern's length back during playback, say from 16 to 8 steps, the sequence keeps running to the end of its old grid before the new length takes effect. That is not a bug but intended behaviour … and it can be used for surprising rhythmic displacements once you have understood it.

SNG IN DETAIL — THE SWING

In the tool row, next to the pattern selection, sits the SWG field with a value from 0 to 9. That is the swing, and it is the smallest intervention with the biggest effect in the whole program. Every odd step comes a little later, and the even step before it becomes longer by exactly as much, so the pair stays exactly in tempo. At 135 BPM a swing of 4 gives a ratio of 9 to 5 ticks; at 9 it goes up to 11 to 3. From three or four a straight beat

starts to groove; from six it turns into a shuffle. The value belongs to the song, is saved with it, applies in pattern mode too and in the export. No other editor on the C64 has this as a single knob (as I know), and after the first listen I did not touch it again for a long time, because I could not believe that this stiff beat was suddenly running.

/// SNG IN DETAIL
THE SWAP POINTS

The two swap rows SWP1 and SWP2 are the arranger's truly powerful tool. A swap point is a spot in the song where a channel changes its instrument. That way you can carry a whole track with a handful of instruments and still keep things varied, without having to set up a new instrument for every bar.

The crucial thing about swaps is that they latch. A swap, once set, switches a channel's instrument, and that instrument then stays in force — across as many bars as you like — until the next swap on the same channel decides otherwise. So you don't set one point per bar, only at the places where something is actually meant to change. That is why this feature gets by on so little memory: a single swap covers an entire passage.

SETTING A SWAP:

You click in one of the two swap rows at the desired spot in the song. That selects the point, and in the selector below you set which channel should switch to which target.

CHOOSING THE TARGET:

The number field in the selector steps the target forwards, a click on the name field next to it steps backwards: the 16 presets one after another, and then the ORIG position. "ORIG" switches the channel back to its original instrument — that is, exactly the setting the channel had at the start of the track, including all the changes you made to it in the editor. So after a change you can always get cleanly back to the starting sound. In addition, you can also step a channel all the way down to zero, meaning no more swap on that channel. Bear in mind, though, that a swap must have a change set in it for it to stay in memory, because otherwise it is simply deleted to save memory.

DELETING A SWAP:

If you click again on a swap point that is already set in the row, it is removed again ... just like a bar in the arrangement. The point disappears, and from there on the channel follows the previous swap again. That is how you build a region with a switched instrument: set a swap to the desired preset at the start, and later a second swap to ORIG that switches back to the original. Everything in between runs with the swapped sound.

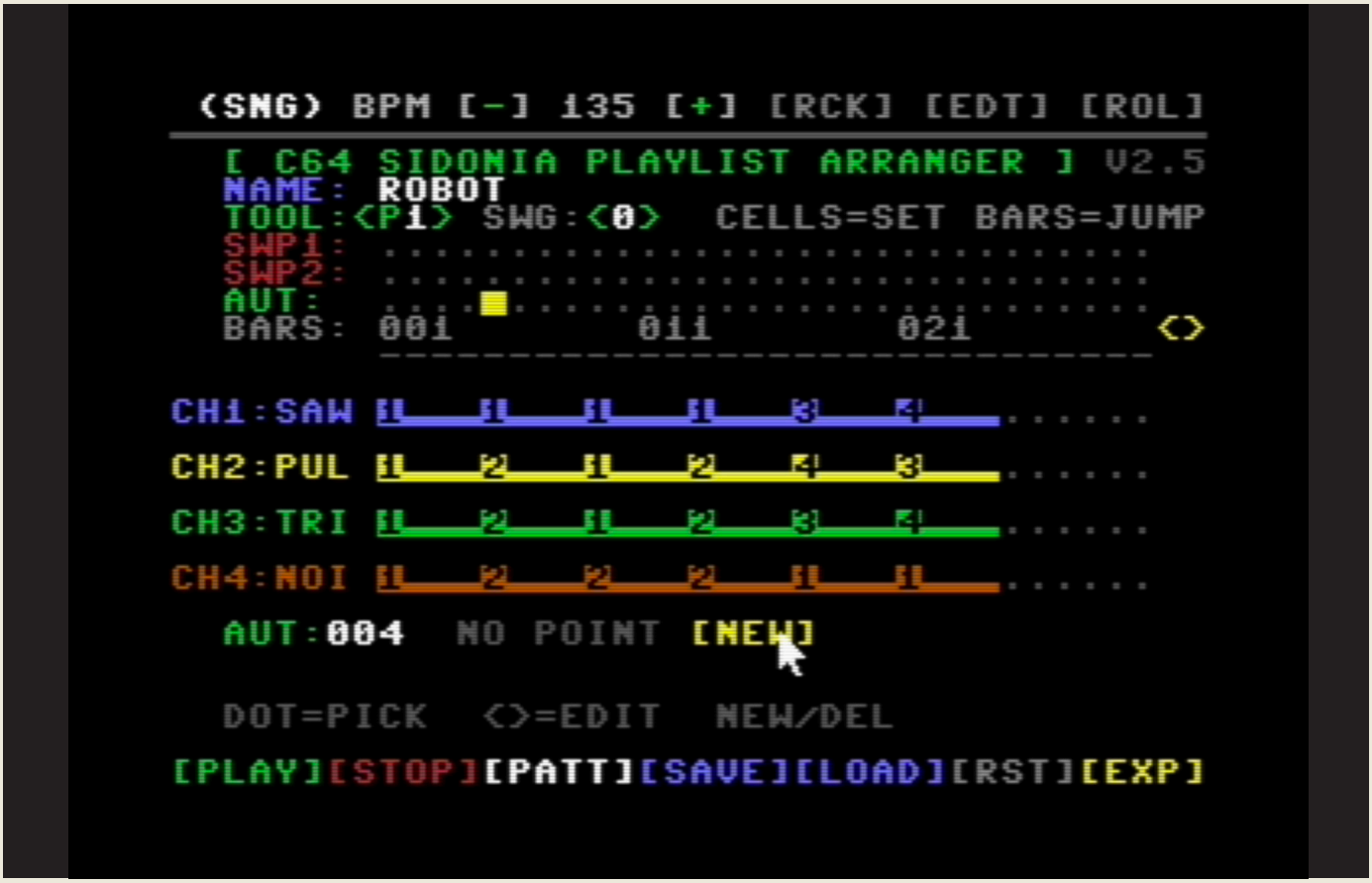
A note on something that can cause confusion while testing but isn't a problem: if you go into the editor during playback while a swap is currently in effect, the editor shows the switched instrument and not the original. The ADSR values then appear to "jump". That is correct ... the editor always shows the sound that is actually on the channel at that moment. If you want to edit an instrument in peace, just stop playback or switch from song mode to pattern mode.

SNG IN DETAIL
THE AUTOMATION

Below SWP2 lies the third line, AUT:, and it is what the swaps were still missing. A swap exchanges a whole instrument. THE AUTOMATION drives a single parameter across the song, the way you drag an envelope across a track in a modern DAW: a filter that opens over four bars, a master volume that fades out at the end, a vibrato that slowly swells in the bridge.

The operation is deliberately the same as with the swaps. A click on a position in the AUT LINE selects it; a yellow grid appears there, and the panel opens at the bottom. If you click somewhere else, the grid disappears again, since nothing was set. [NEW] creates a point at this position, the grid becomes a solid yellow square, and now you use the green arrows to set: CH, the channel; PAR, the parameter; VAL, the value from 0 to 15; CRV, the curve. If you then click elsewhere, the square turns green and stays. Green means set, yellow means in progress, grid means merely looked at. A green point can be clicked and edited again at any time, [DEL] deletes it.

Fourteen parameters are available. The first eight belong to the sound and the level: CUT, the filter cutoff; RES, the resonance; PWD, the pulse width; VOL, the channel volume; VIB, the vibrato; PWM, the speed of the pulse-width modulation; GLD, the glide; and MST, the master volume of the whole chip, with which you can fade a track out cleanly. Then there is the complete envelope: ATK, DEC, SUS and REL can be driven across the song, so a pad that slowly attacks more softly in the bridge, or a bassline whose release opens up towards the finale, is a single point. And two switches: ARP, the arpeggio type from 0 to 4, and FTP, the filter type from 0 to 3. These two are not driven but held, and they jump over at the next point, because there



is no such thing as half a chord type. The envelope takes effect from the next struck note, the way the SID reads it.

A SINGLE POINT SAYS:

from here, this parameter on this channel has this value. If a second point for the same channel and the same parameter comes later, SIDONIA drives the value between the two along the chosen curve, tick by tick, for the cutoff even at a finer resolution than the editor itself can display, so that a sweep really sounds smooth.

THERE ARE FOUR CURVE SHAPES:

LIN, even; EXP, starting slowly and fast at the end; LOG, the reverse; SMO, an S-curve that begins gently and ends gently. On the right of the panel SIDONIA draws the chosen curve as a little graphic, so you don't have to guess.

After the last point the value is held, before the first point the instrument is left untouched. Several points on the same position are allowed, you page through them with PT: < >. In all, a song holds 48 points. That sounds like few, but it is the same logic as with the swaps: a point covers an entire passage, you only set it where something is meant to change.

When jumping back to the start of the song or of a loop, SIDONIA resets the automated parameters to their state at the start of playback, exactly like the swaps. And on stopping as well, so that after a session nothing is left hanging in the instruments that you never set.

/// SNG IN DETAIL
THE LOOP RANGE

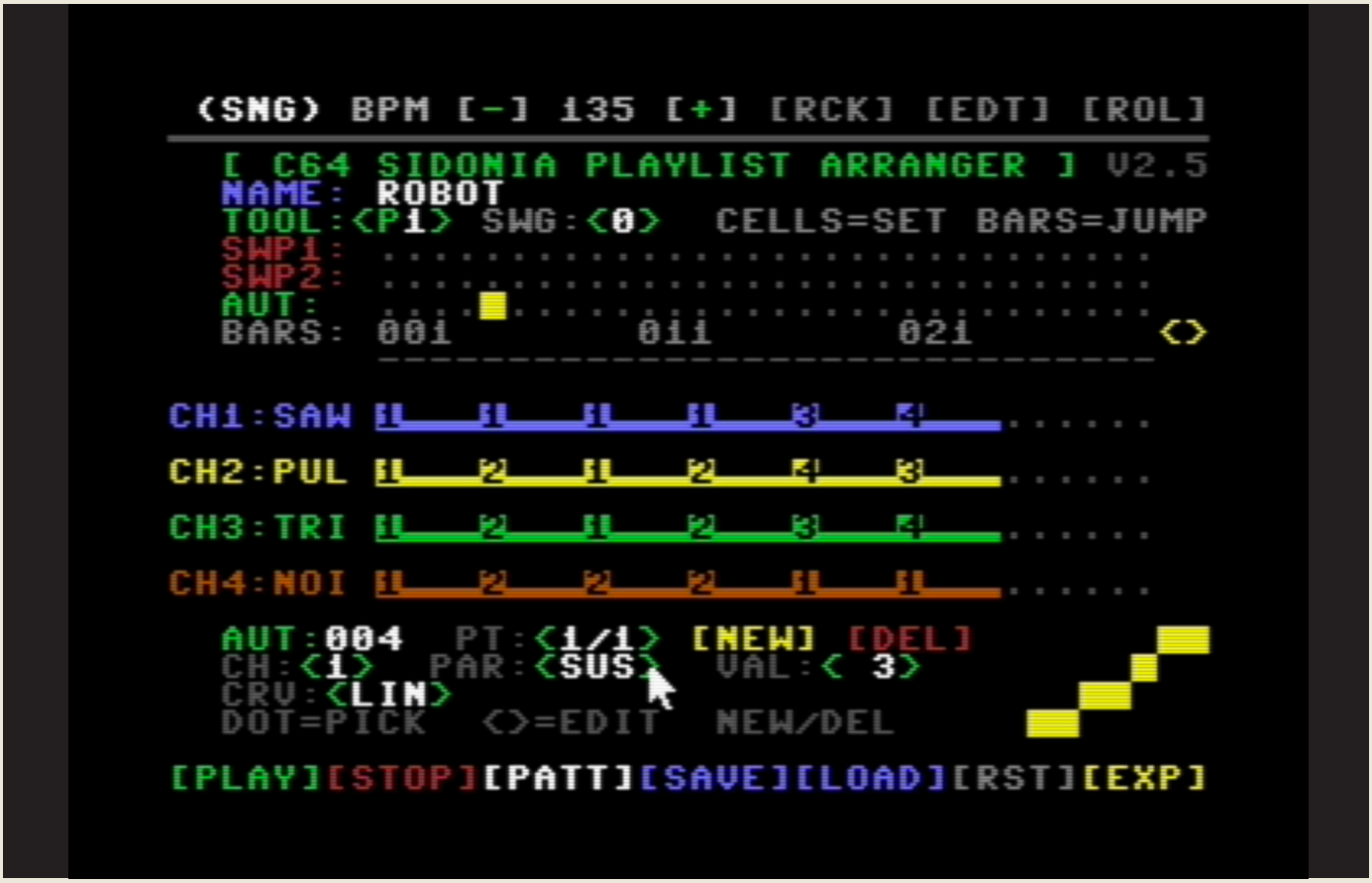
Below the SWAP ROWS lies the bars: A line, on which the song positions are shown. On this line you set the loop range ... a tool that is worth its weight in gold when working, because it lets you listen to a particular section without having to hear the whole track through from the start again and again.

SETTING IT TAKES TWO CLICKS:

The first click on the bars: line sets the start point of the loop. It appears as a green marker. The second click sets the end point. It appears as a red marker. From now on, playback only cycles between these two points. If you accidentally set the end point to the left of the start point, SIDONIA swaps the two by itself so that the start always lies before the end. So you don't have to pay attention to the order.

DELETING THE LOOP:

A click on one of the SET MARKERS cancels the loop again, and the track



runs as a whole once more.

If you press [PLAY] while a loop is set, playback begins at the start point and repeats the range until you stop it. When jumping back to the start of the loop, SIDONIA cleanly resets the instruments to their state at the beginning of the range, so that the loop sounds the same every time. If the cursor runs over one of the markers, the bracket is briefly shown inverted, yellow with a black character, so that you can always see the playhead even under the bracket. That way you can let a transition, a chorus or a tricky passage run round in a loop in peace and polish it.

/// FROM SONG TO FILE
SAVING, EXPORTING, PUBLISHING

Once a track is finished, you want to save it and at some point pass it on. SIDONIA has various file types for this, and it is worth keeping them clearly apart, because they serve quite different purposes.

THE SONG NAME:

Before anything is saved, you give the piece a name in the SNG window. A click on the name line opens the input, RETURN completes it. This name is important, because it later determines what the files on the disk are called ... both when saving and when exporting. If you enter "TRANCE", you get a TRANCE.SNG and, on export, a TRANCE.MUS.

[SAVE] and [LOAD] — the working format. These two commands write and read a file with the extension .SNG. It contains the complete working state: TEMPO, SWING, ALL PATTERNS, THE ARRANGEMENT, the FOUR INSTRUMENTS including ATTACK WAVE, the CHANNEL VOLUMES, the PRESETS and their NAMES, the SWAPS and the AUTOMATION. A .SNG is the format you keep working with in SIDONIA ... you load it back and carry on where you left off. Saving goes to the disk drive with device number 8.

Both commands open a small FILE BROWSER that lists the songs on the inserted disk, and only the songs: the program itself, exports and other files stay out, so that the list stays clear. When loading, you click the desired title. When saving, the current song name is already there; but you can also click an existing entry to overwrite it. Because deleting your own work would be annoying, SIDONIA demands a second click as confirmation in this case. With [CANCEL] you get out again without consequences. And a file that is damaged or doesn't match the format is rejected, without a single byte of the song in memory

being changed. After a night in which a half-written track got wiped out of memory from disk, that was a personal concern of mine. (it still happened)

[EXP] — the export to the finished piece. This is the button that turns your song into a creature that stands on its own two feet. SIDONIA writes a file with the extension .MUS to the disk, and in this file sits not only your music but also a small PLAYER that brings it to life — a stowaway of six kilobytes that knows exactly one job: playing your track, over and over, without complaint. It masters everything SIDONIA itself can do: the SWAPS, the AUTOMATION with all CURVES, the swing, the ATTACK WAVES, the VOLUMES. I tested it and every single register access to the SID, should be the same. What you hear in SIDONIA is exactly what the .MUS plays.

The result is a complete C64 program. You load it at address $1000, type SYS 4096, and from then on it no longer needs SIDONIA. The little stowaway takes over, lets itself be nudged by the computer fifty times a second (should be) and dutifully plays on, whether anyone is listening or not. Here too: the file carries the NAME of your song. "TRANCE" becomes TRANCE.MUS — the export is just as vain as the save.

HERE IS THE PATH AT A GLANCE:

.SNG — your working state, for further tinkering in SIDONIA. (SAVE / LOAD)
.MUS — your song as a standalone C64 program that runs even without you. (EXP)

/// TURNING THE EXPORT INTO A SID FILE

If you want to get your track out among people — into the HIGH VOLTAGE SID COLLECTION, onto DeepSID, or into one of the SID players the collectors of this world use to listen to their treasures — you need the format that is customary there: the .SID FILE. SIDONIA deliberately does not produce it itself, and for a very good reason: a PSID header does not belong on the C64 but on the computer where the publishing happens. It would be like wanting to print the envelope while you are still writing the letter. The step is easy anyway, and on top of that you only ever do it once.

At its core, a .SID FILE is nothing more than the exported .MUS with a short header stuck in front of it. This header — the PSID header — carries the details that every SID player dutifully displays: TITLE, AUTHOR and YEAR OF RELEASE, plus the two addresses at which the player is woken up and driven along. THE SIDONIA PLAYER was built from the very start so that it fits into this format without the slightest contortion. Sometimes something does fall into place effortlessly after all. (Well, that is how I got to do all of it almost twice.)

For the conversion, the program comes with a CONVERTER, in the simplest form I could think of: a single file called SIDONIA_Converter.html. You open it with a double-click in your browser, on the Mac, under Windows, under Linux, on your phone for all I care. Nothing needs to be installed, and nothing gets uploaded; the page only does its arithmetic inside your browser window. The route consists of three steps, and none of them hurts.

First, you export the song in SIDONIA with [exp]. Afterwards the .MUS sits on the disk.

Second, you drag this file into the converter's drop field. If you work with an emulator such as VICE or have an SD2IEC (SD-CARD-DRIVE) on the real machine, you simply drag the whole disk image (.d64) in; the converter picks out the .MUS by itself.

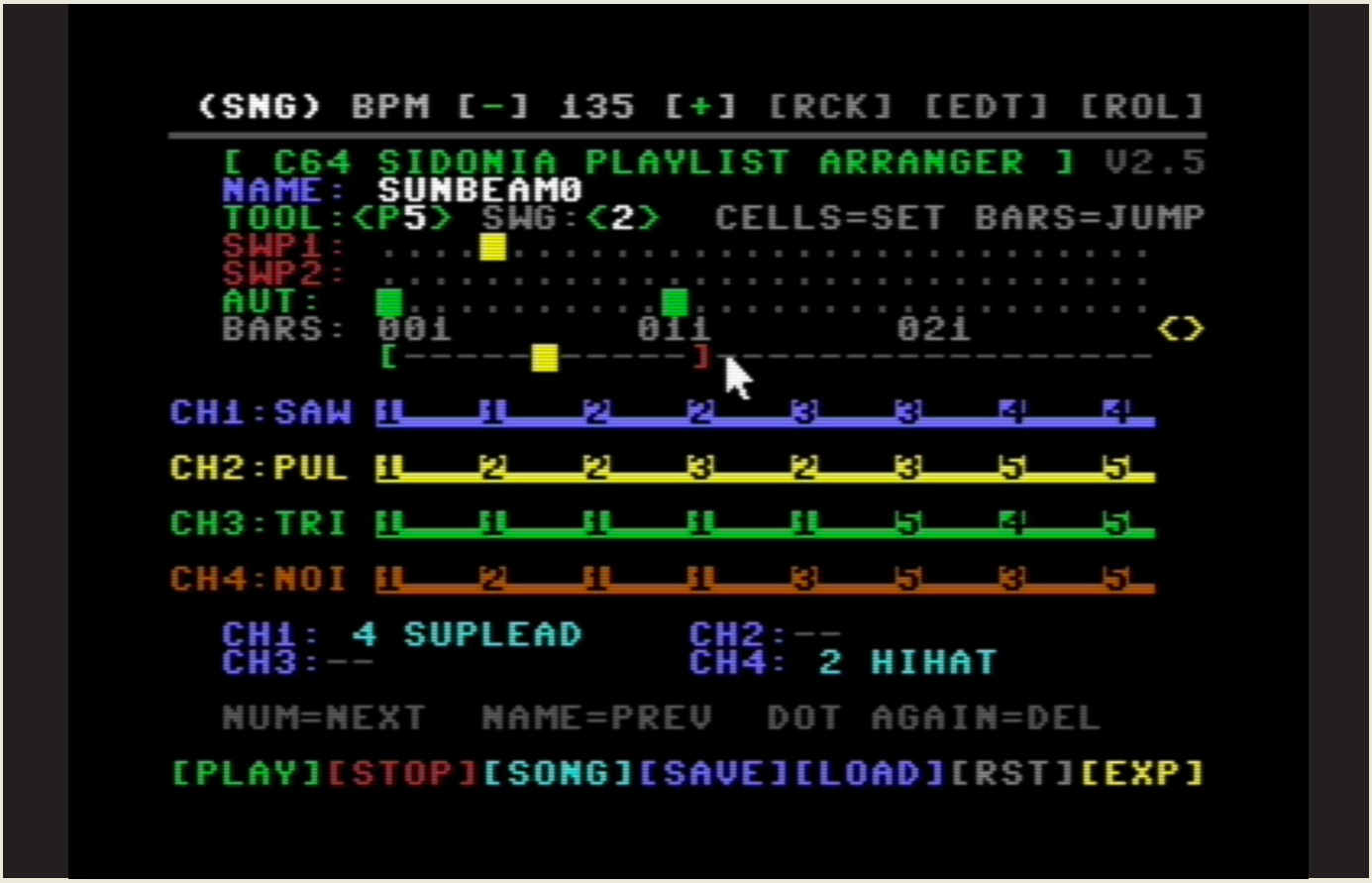
Third, you enter TITLE and AUTHOR and press the button. The .SID lands in your DOWNLOAD FOLDER. TITLE, AUTHOR and YEAR may each be up to 31 characters long; that is the limit the format dictates. The author is mandatory, and it means YOU: the person who built the song, not the program!

That was all. Out comes a valid .SID that you can check straight away on DeepSID in the browser or with sidplayfp, and then send off into the wide world of SID collections. If you prefer working on the command line, you will also find mus2sid.py, a Python script with exactly the same result, byte for byte.

/// BUILDING A SIDONIA TRACK INTO YOUR OWN GAME

Maybe you do not just want to listen to your track but lay it as music under your own GAME or your own DEMO. That is precisely what the export is made for, because the exported .MUS is not a mere morsel of sound but a complete little program with the player already on board. So you do not have to hunt for someone else's player and fiddle anything together; you already have everything you need.

The PLAYER knows exactly two entry points, and both lie at the start of the loaded block. The song is loaded to address $1000. Right at the front, at that very address, sits the initialisation, and three bytes behind it, at $1003, sits the playback. The rest is habit if you have ever built SID music into a



program before: you call the initialisation exactly once, and after that the playback exactly once per frame. In assembler it looks like this. Once at the beginning, when your game starts and the music is supposed to begin:

```
jsr $1000   ; Init: prepare the song, call once
```

And after that, a single time in every frame, that is fifty times per second:

```
jsr $1003   ; play one frame of music
```

The usual and cleanest place for this second call is your raster interrupt routine, which runs fifty times per second anyway. That is exactly where the music tick belongs; then the song keeps running in the background while your game code does its own work. If you do not want to set up an interrupt of your own, you can also make the call from the main loop, as long as you make sure it really comes once per frame, because the tempo depends on it. If it comes more often, the music runs too fast; if it comes less often, it drags.

A few things that will spare you grief. The song sits fixed at $1000, so make sure your own code and your graphics data do not overwrite that area. The player occupies six kilobytes from that address, the song data follows on at $2800 and, depending on the track, reaches up to about $4C00. On the zero page the player uses the addresses $02 to $05 and $FB to $FE; if you keep your own pointers there, put them somewhere else. The SID itself is a single resource that exists only once, so your game cannot output its own sound effects on the same voices at the same time without quarrelling with the music. If you need effects, you have to reserve a voice for them and keep it free in your song.

And if your program bends interrupts of its own, take care that the music tick does not get lost in the process.

From the computer, without a game of your own around it, you can check the result directly at any time. Load the .MUS to $1000 and start it with SYS 4096, which is the same address written in decimal. The player then takes over by itself and plays the track in a loop, and you hear exactly what will later sit underneath your game.

/// WHAT GETS SAVED ALONG WITH IT

A word to everyone who lovingly tweaks their sounds, gives them NAMES and talks to them as if they were little pets: nothing is lost. When saving as well as when exporting, SIDONIA stores the complete state — not just the notes, but also the four

INSTRUMENTS of the main voices, ALL SIXTEEN PRESETS with their tweaked VALUES and, yes, even the NAMES you gave them in a fit of tenderness or despair. Plus the CHANNEL VOLUMES, THE SWING, the ATTACK WAVES and every single AUTOMATION POINT. That is exactly why the SWAP POINTS and the AUTOMATION also work in the finished track: they reach into INSTRUMENTS and PARAMETERS, and these travel along in the file. If you shape an instrument by chance or by hand and give it a name, you will find everything unchanged the next time you load — and in the exported piece it sounds precisely the way it was built. The little passenger simply takes along everything you entrust to it.

/// ON THE LIMITS OF THE MACHINE

One thing should be named honestly, because it is part of the program's character. SIDONIA stores a track as an arrangement of references: each PATTERN lies in memory only once, and the four tracks of the arrangement merely point to it. That is the reason why whole tracks are possible at all on a machine with less than 64 kilobytes — not every note is stored individually; it is only noted which pattern is to sound when. A crib sheet instead of a novel.

This arrangement holds 512 SLOTS per track. At a typical tempo that comes to a track of roughly three and a half to four minutes. That is the one limit you run into, and I thought long and hard about whether I should move it. The space between the data area and the character set is tight and measured out exactly; every shift risks that something elsewhere, which already works cleanly, starts to wobble — anyone who has ever got a C64 to serve all its registers nicely at the same time knows that you do not nudge a running clockwork out of curiosity. I decided against it. Four minutes is a considerable length for a SID track, and the limit is an honest expression of what this machine works with. You do not have to tear down every wall you run into. Sometimes it is the wall that gives the room its shape.

And because that sentence was put to the test more than once in the final phase, one more number to go with it: when the AUTOMATION was added, memory was so tight that the main code had grown a few hundred bytes into the song data without any tool so much as grumbling. The presets were suddenly called "ELK!C" and "KICK" with a tail of garbage behind them. It was not a data error; it was a program that had trodden on its own feet. Since then the build explicitly has a critical boundary, and the code is spread across free areas that the C64 still had left over under its BASIC ROM and between the data blocks. It fits exactly. Interesting and absurd at the same time that the space was precisely enough to accommodate all of it.

/// OUTLOOK

SIDONIA is finished in the sense that it does exactly what it is supposed to do: it opens up the SID CHIP to people who do not want to read HEX CODE, and it does so with an interface that admittedly looks raw, but wants to be understood and can be understood. The TITLE TRACK of this issue was created entirely within it.

This is exactly where art meets technology, and on a machine that gives nothing away for free. Every function in this program is a true compromise with the hardware, negotiated in clock cycles and bytes. The SID has three voices, so two of the four channels share one of them. Memory is tight, so the SWAPS snap into place and the song is a reference to itself. The editor must not compute for too long per frame, so the whole interface runs in text mode. That is not deprivation, that is the form. Anyone who works on this computer does not hack their way through the limits but builds with them.

The result is an audiovisual collision in the best sense: a tool that is INSTRUMENT, EDITOR and PLAYER at the same time, built on a chip that was actually meant for game noises, is full of quirks and has nevertheless been making music for forty years.

The program is available for free download in version 2.5. Take it, build with it, break it, find the corners I did not think of. The SID gives back the most when you force it into something it was never intended for. Make tracks out of it, make soundtracks, make noise. And if you find something that works even better, then you now know that this machine can be moved; you just have to know where to push. And which registers to pull in the end.

Please note that, depending on the hardware, the software may behave more or less smoothly. That is why I clearly recommend testing different emulators, if you use any, and doing so on different machines.





TOP: SAVE SONG. Give the song a NAME, press RETURN, and SIDONIA writes the pattern data, the sixteen presets, the names and the swaps to drive 8. A click on an existing name picks it up again. (Shown: saving "SUNBEAM0" to an empty disk.)
BOTTOM: THE CONVERTER. The browser-based SIDONIA → .sid converter wraps your exported .MUS in a PSID header for DeepSID, the HVSC and any SID player. It runs entirely in your browser; nothing leaves your computer. AUTHOR and TITLE are typed in by you – THE COMPOSER – nothing is filled in by default.



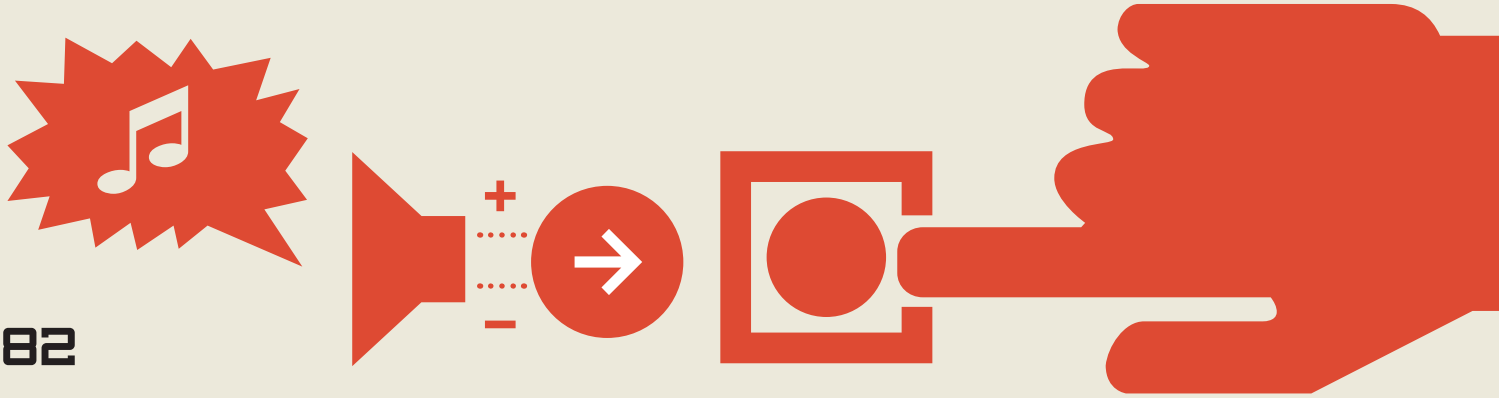


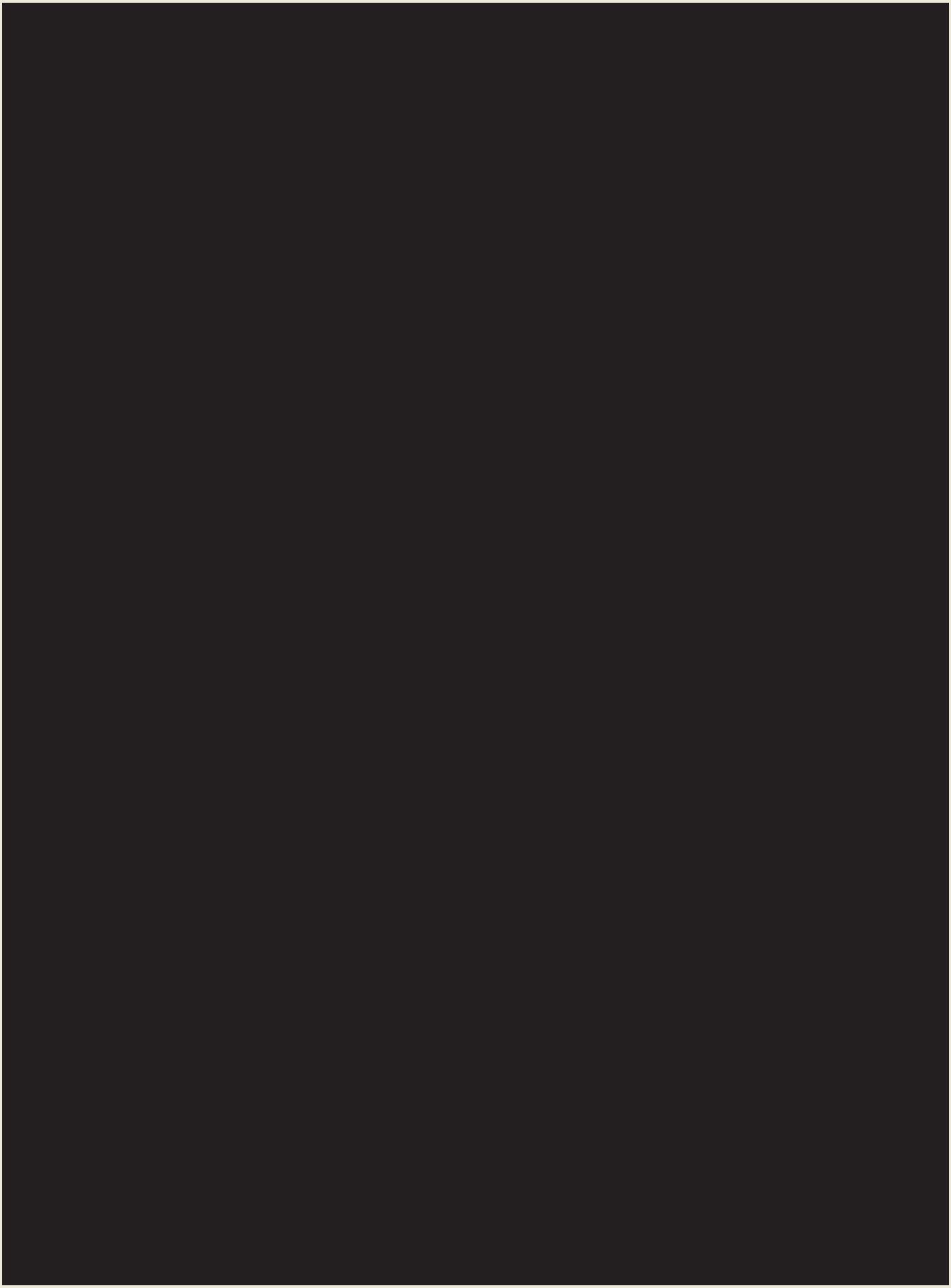
Y OR Z? The labels refer to the C64's own keyboard. If you run SIDONIA in an emulator with a German PC keyboard, Y and Z change places – and with them the C of the lower row and the A of the upper row. There was no room left in memory for a swap option, so please don't be surprised.

PLAYING SIDONIA FROM THE KEYS

The C64 keyboard is your piano, laid out the way every tracker since the eighties has done it. The lower row Z S X D C V G B H N J M plays the twelve semitones of one octave, the upper row Q 2 W 3 E R 5 T 6 Y 7 U the same twelve notes one octave higher. The letters sit on the keyboard exactly as the keys sit on a piano: the naturals in the long rows, the sharps on the row above them – S, D, G, H, J and 2, 3, 5, 6, 7 are your black keys.

Which octave you are playing is decided by the o field in the piano roll: the lower row is octave 0, the upper row 0+1. Change o with o: < > or with the [+] and [-] fields beside the keyboard column, and both rows move with it. The setting is global, so the keys play the same pitches in RCK, EDT and ROL. What you hear is always the channel you are currently working on, with its instrument and its own octave from the editor on top. Whenever a note is played, the marker column beside the PIANO ROLL flashes the key and its note name, so you can see on the screen which key you just hit.





JOAN RUE AT THE BENCH. /// SIDONIA v2.5 running on the original hardware: a Commodore 64 with 1351 mouse, SID 6581, autumn 2026. Every screen in this manual was tested here before it went out to the world.