# synth secrets

## PART 1: WHAT'S IN A SOUND?

In the first part of this new series exploring the world of subtractive synthesis, Gordon Reid goes right back to basics. What are waveforms and harmonics, where do they come from, and how does the theory relate to what we actually hear?

Despite the title of this article, I won't be revealing any actual secrets during the course of this new series. But before you turn the page in disgust... what I will be doing is taking a look at the basic principles of that most common form of sound synthesis, subtractive synthesis (these principles are well known, and therefore hardly secret), and, later on in the series, how these principles are applied on specific synthesizers. The aim is that if you have a synth that works on subtractive principles, and you know how to get sounds out of it that you like, but don't understand *why* or how it makes the sounds it does, this series should fill in some of the blanks (they're the 'secrets', you see). OK, maybe we should have called the series Why Your Synth Does What It Does When You Twiddle That Knob Or Slide That Slider... but, let's face it, that's not exactly a catchy name. So Synth Secrets it is. First things first, then: what *is* subtractive synthesis?

The name 'subtractive synthesis' is derived from the method itself, wherein you attenuate or remove harmonics from harmonically rich waveforms to create new sounds. You can do this in a static fashion to create simple tones, or you can use the facilities offered by the filters, envelope generators, and modulators in your synthesizer to make dynamic sounds that change as time passes. But... you may be lost already. What actually are harmonics? What are waveforms? Where do they come from? This month, I'm going *right* back to basics, and answering just those three questions. The stuff about VCFs, EGs and LFOs will come later.

### It's All Greek To Me

To answer these fundamental questions, we have to jump into the *Sound On Sound* time machine (it's tucked away somewhere behind the photocopier) and head off into the dim recesses of the past. Back before physical modelling, before samplers, before analogue polysynths, even before monosynths...

Actually, we're in serious *Dr Who* territory here, because we need to head back 2,500 years and reacquaint ourselves with an Ionian chap by the name of Pythagoras. Pythagoras was perhaps the world's first pure mathematician, yet we know relatively little about him or his achievements (much of what we do know about him may be no more than legend — in contrast to what every schoolboy knows, the Babylonians discovered Pythagoras's theorem about 1,000 years before Pythagoras was born).

One of the lesser-known discoveries attributed to Pythagoras was that plucking two similar strings stretched to the same tension gave a pleasing sound if their lengths were related by simple integers (ie. whole numbers). For example, if one string was half the length of the other (a 1:2 relationship) the result sounded quite nice. If the relationship was 2:3, that sounded pleasant too.

Pythagoras was blown away by his discovery, and placed numerology at the heart of his philosophy. Unfortunately, he and his followers then went off the rails a bit and tried to determine similar numerical relationships for the periods and orbits of the five known planets, the sun and the moon, thus giving rise to the mythical 'music of the spheres'. If they had only looked at the very small instead of the very large (discovering Quantum Mechanics in the process) they would have been much more successful.

But why did Pythagoras's strings have integer relationships? Why weren't similar, pleasing sounds generated by two strings when one of them was 1.21346706544 times the length of the other?

### Let's Get Plucking

To start answering this, let's consider a stretched string that is fixed at both ends, but free to vibrate along its length. Figure 1 shows such a string at rest.

Now imagine that we gently pluck the string exactly halfway between the ends. As you might imagine, this causes it to vibrate in the way shown in Figure 2.

This is an example of a 'standing wave'. It does not run up and down the string like waves on the surface of the sea, but vibrates up and down. If the vibration (or 'oscillation') is as simple as that shown in Figure 2, a point at the centre of the string moves in a simple, repeating pattern called a sine wave (see Figure 3). We call this pattern the oscillation's 'waveform', and the frequency with which the waveform completes one 'cycle' is called the 'fundamental' frequency of the string.

The fundamental mode is, however, not the only way in which the string can vibrate — although because it is fixed at both ends, the number of ways and the speeds with which it can do so are severely

Figure 1: A simple stretched string.

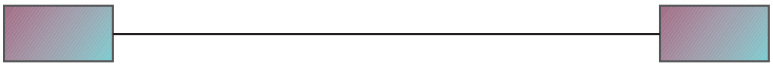
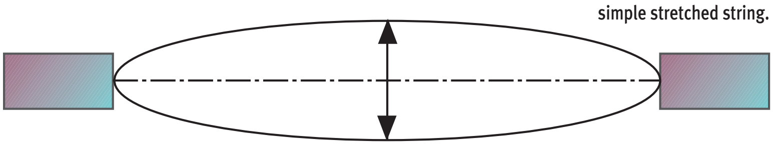


Figure 2: A vibrating simple stretched string.



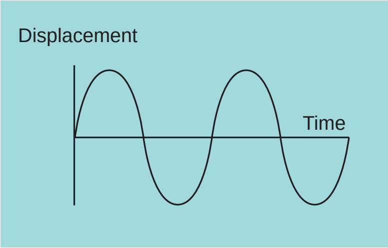


Figure 3: A sine wave.

constrained. Imagine placing your finger in the exact centre of the string (but so that the string can still vibrate along its entire length) and plucking it on one side or the other. You can see from Figure 4 that a standing wave with half the wavelength of the original looks perfectly feasible.

Likewise, if you place your finger one-third of the way along the string, a standing wave of one-third the wavelength of the original should be possible (Figure 5) and so on.

Indeed, these standing waves can exist at all the integer divisions of the wave shown in Figure 2, and we call these the 'harmonics' of the fundamental frequency.

If you study the maths of standing waves, you'll discover that you can represent such a wave as two 'running' waves moving in opposite directions along the string (no, don't ask why, or we'll be here until page 304). Knowing this, however, leads us to a simple conclusion: if you halve the wavelength, the frequency of the 'running' waves required will double. Similarly, if you divide the wavelength by a factor of three, you triple the frequency; quarter the wavelength and you multiply the frequency by four, and so on... Only whole numbers will work because, if you tried to introduce a non-integer change in the frequency, the string would need to be somewhere other than at the zero position at one of its ends (ie. part of the way through an complete cycle), and this isn't possible because, of course, the ends are fixed.

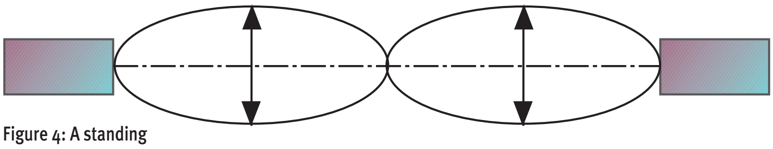


Figure 4: A standing wave with one-half the wavelength of the fundamental.

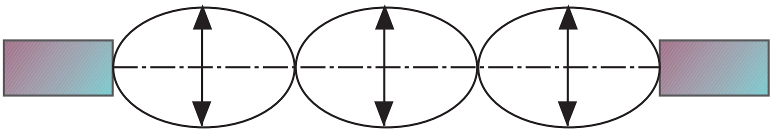


Figure 5: A standing wave with one-third the wavelength of the fundamental.

Anyway, we've now answered our first question by identifying the harmonics which can be produced by a simple oscillator: they are the permissible modes of vibration. Of course, this analysis doesn't only apply to a vibrating string. Consider the air in an enclosed space such as a cubic room. Forgetting for a moment any complicating factors such as furniture, the air can vibrate anywhere in the room except at the walls, floor, and ceilings. In other words, the vibrations in the room are constrained in the same way as those on a string. This is why regular rooms have 'resonances' — they are the harmonic frequencies of the room itself. And this is why cathedral organs work — pipes are also simple harmonic oscillators.

In all but some esoteric cases, the first harmonic (the fundamental, called *f*) is the pitch that you'll perceive when you listen to the sound of the plucked

"The name 'subtractive synthesis' is derived from the method itself, wherein you attenuate or remove harmonics from harmonically rich waveforms to create new sounds"

▶

string. The second harmonic (also called the first 'overtone') is half the wavelength of the fundamental and therefore twice the frequency. In isolation we would perceive this as a tone exactly one octave above the fundamental.

The third harmonic has a frequency of 3*f* (which is the perfect fifth, one and a half octaves above the fundamental) and the fourth harmonic, with a frequency of 4*f*, defines the second octave above the fundamental. The next three harmonics then lie within the next octave, and the eighth harmonic defines the third octave above the fundamental. And so it goes on...

This is the information we need to understand Pythagoras's observation. The shorter of the two strings in the 1:2 relationship is producing a fundamental at the same frequency as the second harmonic of the longer one. It's exactly one octave higher. In the example of the 2:3 strings, the third harmonic of the longer string is at the same frequency as the second harmonic of the longer one. In other words, the harmonic structures of the two strings are closely related to one another, and we hear this as musically 'pleasing'.

## The Nature Of A Sound

Now consider this: when you pluck a string, you don't hear the sound of a single harmonic. The conditions for creating such a pure tone are — in the real world — almost impossibly precise, so any naturally occurring tone is likely to be a composite of many harmonics present in differing amounts. At any given moment it is this combination that determines the waveform of the sound and, because of the number of harmonics present, this waveform will be much more convoluted than the simple sine wave shown in Figure 3. You only have to look at a sample of a guitar or a human voice in a waveform editor to see how complex a real waveform can be.

This would make analysis of sound — or its resynthesis — almost impossibly difficult, had it not been for a French mathematician named Jean Baptiste Joseph Fourier. Another guy with a colourful life, Fourier was in turns a teacher, a secret policeman, a political prisoner, governor of Egypt, Prefect of Isère and Rhône, and a friend of Napoleon. Despite this, he still found time to determine that any periodic motion, no matter how complex, could be broken down into its harmonic components. This procedure is called Fourier Analysis in his honour. Furthermore, Fourier analysis also shows that, given a set of harmonics, you can derive a unique waveform.

Hold on a second... the waveform defines the harmonics, and the harmonics determine the waveform? Clearly, harmonics and the waveforms are just two ways of expressing the same thing. This is a key point: the natures of musical tones are defined by the numbers and amplitudes of the harmonics contained within them, and any given set of harmonics gives us a given waveform. So when we look at the oscillators on a synth and see things such as 'square' waves or 'sawtooth' waves, this is simply a shorthand way of saying, "this setting generates a particular set of harmonics with amplitudes of x, y and z..."

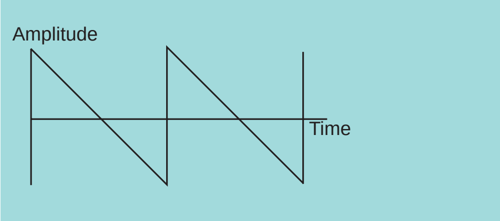


Figure 6: The simple sawtooth wave.

## Subtractive Synthesis

So let's apply these ideas to a synthesizer. Look at the waveform in Figure 6. You would never get this from a plucked string, but you'll find an approximation to it on almost every synthesizer ever built. It's an ideal 'sawtooth' wave, so named because of its shape.

This waveform has a simple harmonic relationship, expressed as follows:

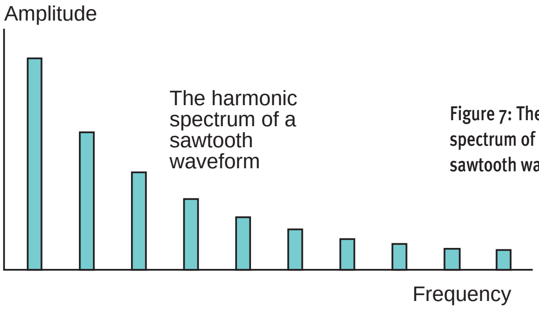


Figure 7: The harmonic spectrum of a simple sawtooth wave.

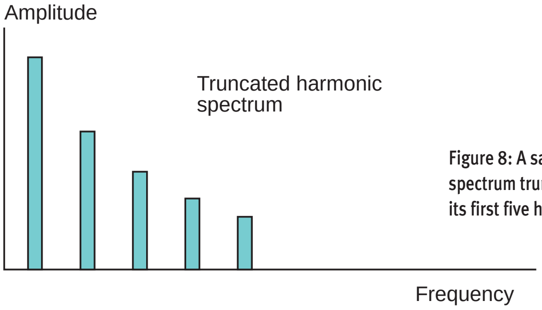


Figure 8: A sawtooth spectrum truncated to its first five harmonics.

*Every harmonic is present, and the amplitude of the nth harmonic is 1/n times that of the fundamental.*

OK, so it doesn't look so simple when written in English but, believe me, there are far nastier ones than this. Anyway, Figure 7 shows the first 10 harmonics in a sawtooth wave, and you can see how they taper off at higher and higher frequencies.

But what happens if you truncate this series of harmonics? Let's say you remove all but the first five of them (for which you need a device called a 'filter'). Figure 8 shows this spectrum, and Figure 9 shows the waveform to which it corresponds.

As you can see, the new waveform looks different from the sawtooth wave. It sounds different too. But the only difference between them is that you have truncated the harmonic series of the sawtooth so that only the first handful of harmonics remain. In other words, you have used a 'filter' to 'subtract' harmonics, thereby creating a new waveform, and thus a new sound.

Welcome to the world of subtractive synthesis! SOS

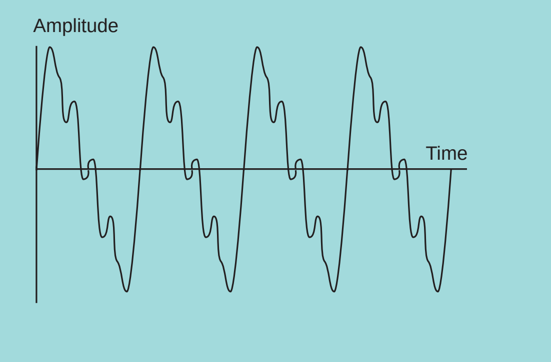


Figure 9: The waveform created by the first five harmonics of the 1/n series.

# synth secrets

## PART 2: THE PHYSICS OF PERCUSSION

The first part of this series explained how the tones of most real instruments can be reduced to patterns of harmonics, which can be generated using sine, saw, square or pulse waveforms. This month, Gordon Reid considers the sonic raw materials needed to imitate unpitched percussion.

Last month we posed the question, "What are harmonics, and where do they come from?" then answered it by considering the way in which sounds can be broken down into, and built up from, their constituent parts. We used a vibrating string as the main example of a harmonic oscillator and, in doing so, described the fundamental properties of every instrument from a double bass to an electric guitar, taking in pianos, harps, violins, zithers, and bazoukis on the way.

The properties we discussed are just as applicable to the other major groups of musical harmonic oscillators, such as the blown pipes that comprise the organ, woodwind and brass families. Consequently, as we'll see in future instalments, it is possible to imitate (or 'synthesize') many 'real' instruments using a very small number of simple harmonic waveforms. For example, a sawtooth wave will provide the basis of most of the orchestra's brass and string sections, a square wave will synthesize woody sounds such as clarinets, while the thinner-sounding pulse waves provide the reedier tones of oboes and bassoons.

Moving away from the orchestra, you can use the same waveforms to synthesize modern instruments. A combination of sawtooth and pulse waves provides some remarkably accurate imitations of bass guitars and, suitably modified, a sawtooth will generate a range of timbres that you can play through effects units to sound just like a lead guitar. Of course, you have to *play* the sound like a guitar too, but that's another story. The important point is this: a synthesizer that offers just three waveforms provides you with the raw materials to imitate most of the instruments that you'll find played by an orchestra, a rock group, and the Viennese Oom-Pah Champions of 1898.

But there is an important class of musical oscillators that do not fit the simple harmonic model. These 'non-harmonic' oscillators are just as important as their harmonic cousins, but they do not conform to the same set of rules. Examples of these include drums, timpani, and many of the ethnic instruments now used as sound effects to spice up western music. So why do these sound different and, more importantly, how are we going to get our subtractive synthesizer to imitate them?

### A Multi-dimensional Problem

Consider for a moment the stretched string that we discussed last month. Ignoring its negligible diameter and the carved lump of resonating wood and metal bits that usually come with it, this has one dimension... its length. Other properties, such as its density and its tension, affect it, but in a three-dimensional universe it only has one primary dimension. The same is (approximately) true of a pipe. The nature of its bore significantly affects it, as do factors such as the material from which it is made, but again, its single most important dimension is that of length.

Now picture a circular membrane that is stretched with an equal tension at all points, and which is fixed at all points around its circumference. You would normally call this a tambour or a drum skin, but we are going to think of it as a different type of oscillator. Again, we can ignore complicating factors such as the bits of wood and metal that accompany it (the shell), and concentrate on the oscillator itself.

The most important difference between a membrane and a stretched string is the one that is plainest to see: unlike the string, a drum skin has two dimensions — it is a surface, rather than a line. Consequently, you might guess that drums would respond very differently to being struck, plucked, blown, or whatever it is that you do to excite them. And you would be right. The number of physical dimensions possessed by an oscillator is instrumental (oops, sorry) in determining its acoustic nature.

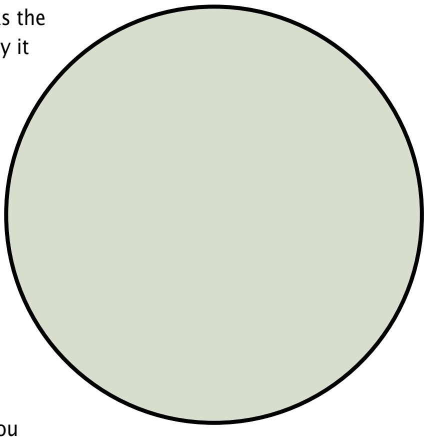


Figure 1: The stationary drum skin.

▶

## The Banged Drum

Let's consider a stationary, circular drum head. Like the string described last month, this is fixed at its 'ends'. Or, to be precise, it is fixed all around its circumference, as shown in Figure 1 on page 60. And just like our string, the drum skin is not free to move up or down at the points where it is fixed.

Now imagine that you hit the drum skin exactly in its centre. You might expect it to move up and down in a single motion, just like the fundamental of the vibrating string. And you would be right again. Viewed from its side, the fundamental frequency of a vibrating circular membrane looks suspiciously like that of a vibrating string (see Figure 2). This, for reasons we need not go into, is called the *w01* mode of the membrane.

(At this point you can breathe a sigh of relief because, unlike last month, I am not going to make you worry about any maths. This is not out of respect for you, but because the equations relating to a vibrating membrane are capable of giving astrophysicists serious headaches.)

Since you are hitting the drum skin in its centre, you can't put your finger in the middle as you did to create the second harmonic of the string, so let's look at the equivalent of the third harmonic. If you remember last month's article, you'll recall that you put your finger one-third of the way along the string to create an overtone of exactly three times the frequency of the fundamental. But if you try this with the drum skin, placing your finger one-third of the way from the centre to the rim, you're in the wrong place. Instead of having 'zero points' described by a simple 1/(integer) relationship, the drum's zero points are described by a hideous equation called a Bessel Function. This tells us that the first zero point is 42.6 percent of the distance from the centre to the rim. What's more, the frequency of a drum skin vibrating in this way (called the w02 mode) is 2.296 times the fundamental. So, while the 'odd' overtones of the string and the membrane can look similar, their musical properties are very different (see Figure 3).

> **"My advice is this: don't even think about trying to analyse the sound produced by something that, at first sight, seems as simple as a drum skin."**

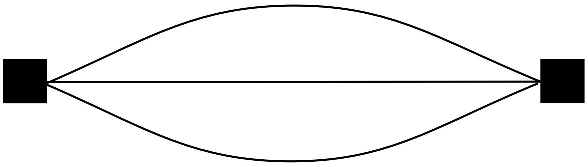


Figure 2(a): The fundamental of a vibrating string.

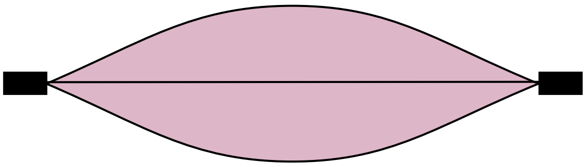


Figure 2(b): The WO1 mode (fundamental) of a vibrating circular membrane.

And so it goes on... The next odd harmonic of the vibrating string has five equally spaced sections, and oscillates at exactly five times the fundamental frequency. The equivalent for the drum skin (the w03 mode) has zero points at 27.8 percent and 63.8 percent of the distance from the centre to the rim, and it oscillates at a frequency of 3.6 times that of the fundamental (Figure 4).

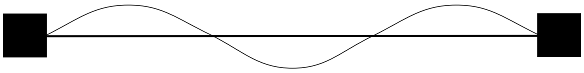


Figure 3(a): The third harmonic of a vibrating string.

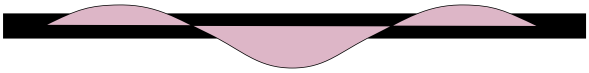


Figure 3(b): The Wo2 harmonic of a vibrating circular membrane.

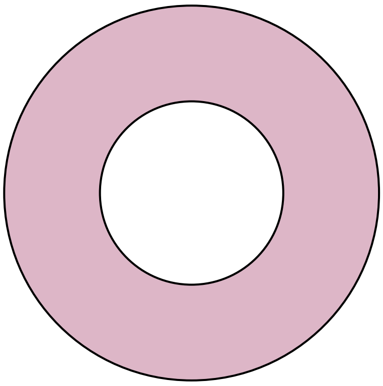


Figure 3(c): The Wo2 harmonic of a vibrating circular membrane viewed from above. When the white centre is 'up', the shaded area is 'down' and vice versa. The black lines are the 'zero points' where the membrane is unmoved.

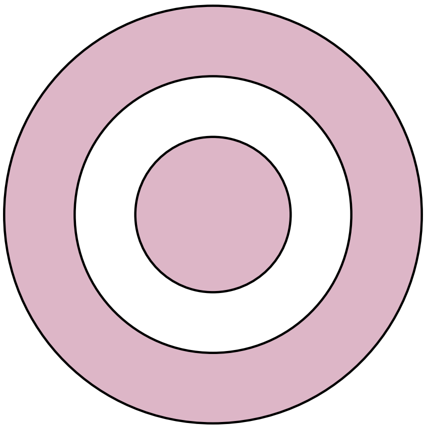


Figure 4: The W03 harmonic of a vibrating circular membrane viewed from above. When the white areas are ‘up’ the shaded areas are ‘down’ and vice versa.

Just to make matters complicated, the drum skin vibrates in completely different ways if you do not strike it precisely at its centre (which, in the real world, is always). Figure 5 shows a small selection of these other modes and their relationships to the fundamental frequency, f.

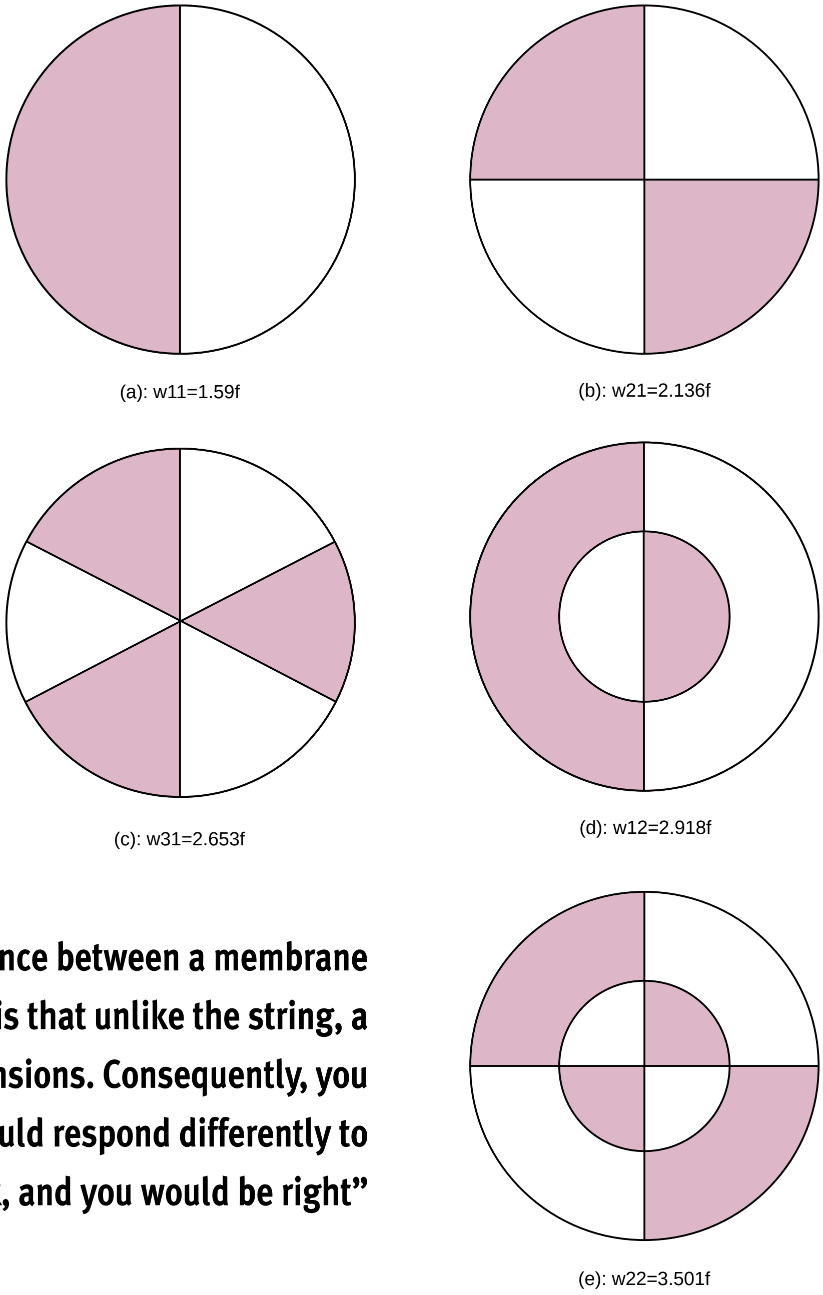
Just like a vibrating string, the drumhead is almost always excited in such a way that a number of its different modes oscillate simultaneously. Unfortunately, they will all have different amplitudes, and all decay at different rates. This makes the drum’s sound enormously complex and — here’s the important thing — impossible to emulate using the types of waveforms produced by a simple harmonic oscillator.

To explain this a little more clearly, let’s look at the position of the first four harmonics of the sawtooth waveform, and compare them to the first few harmonics of the drumhead (see Figure 6 on page 66). As you can see, the drum skin generates more harmonics, and they are clustered in an uneven manner, unlike the regularly spaced overtones produced by the simple harmonic oscillator. This makes the sound ‘atonal’, and stops us from perceiving a simple pitch and tone. Indeed, if you look beyond these first few harmonics, you’ll find that the drum skin’s overtones become more and more numerous and more closely spaced. If we drew the same diagram as 6(b) for a drum skin with a fundamental of, say, 100Hz, but extended the frequency axis to 20kHz, it would look like an inseparable bunch of harmonic frequencies extending right up to (and beyond) the limits of hearing.

If all this is beginning to make you boggle, you won’t want to consider the further complications that exist in the real world. For example, no matter how carefully you adjust it, a drum skin will always have slight variations in tension across its surface, so the modes will be distorted and, in all likelihood, impossible to calculate. And every drummer knows that, when you hit a drum harder, its pitch rises. This means that the fundamental frequency is in some way related to the displacement of the membrane. Aargh!

My advice is this: don’t even think about trying to analyse the sound produced by something that, at first sight, seems as simple as a drum skin. Research groups have spent decades creating

Figure 5: Some of the ideal drumhead’s more complex permitted modes of vibration.



(a): w11=1.59f

(b): w21=2.136f

(c): w31=2.653f

(d): w12=2.918f

(e): w22=3.501f

“The most important difference between a membrane and a stretched string is that unlike the string, a drum skin has two dimensions. Consequently, you might guess that drums would respond differently to being struck, and you would be right” hugely sophisticated mathematical models of vibrating membranes and, as you know, the few DSP-based products that use these are still distinguishable from the 'real' thing. So is it time to admit defeat and consign our analogue synthesizers to the ignominy of 1970s-style trumpet and 'cello imitations? Strangely, no...

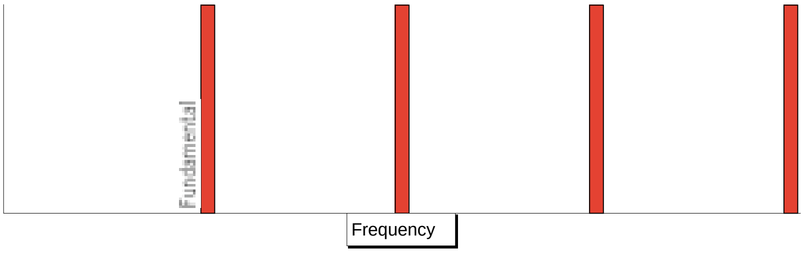


Figure 6(a): The fundamental and first three overtones in the spectrum of a simple harmonic oscillator.

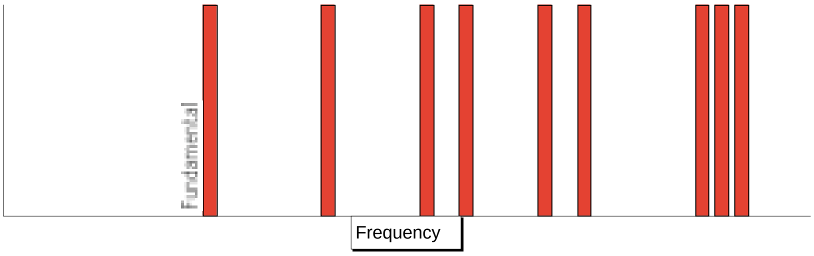


Figure 6(b): To the same scale, the fundamental and first eight overtones in the spectrum of a drum skin with the same fundamental frequency as 6(a).

## Synthesizing The Un-synthesizable

Let's think about 'drum' sounds in the context of an analogue synth. If we are to produce a convincing imitation, we need to generate the dense cluster of frequencies described above, and ensure that they are not, in the conventional sense, harmonically related. Fortunately, most synthesizers have a module that produces something similar. A perfect 'noise generator' produces all audio frequencies simultaneously, and this is close enough to provide a basis for many analogue 'drum' sounds that would be quite unobtainable using conventional waveforms (see Figure 7). Indeed, filtered noise is the basis of the most popular percussion sounds of the 1990s — the Roland CR78, CR5000, TR808 and parts of the TR909 all generate their voices in this fashion.

But what about other circular percussion instruments that are not stretched in the same way as a drum skin? In many ways these instruments — including cymbals and gongs — are very similar to drums. Of course, they are rigid and they are not fixed at their edges, so they are free to vibrate in different ways. But their fundamental natures are governed by the same maths as that describing the stretched membrane. Even bells (which look 3-dimensional because they occupy a volume) are better described as 2-dimensional oscillators because they are, in essence, bent sheets.

Unfortunately, the physical differences between drums and metallic percussion instruments mean you can't program convincing bells and gongs using a noise generator. Consequently, this is an area in which a basic synthesizer such as the Minimoog in figure 7 does not excel. But if you look at the patch sheets of a more complex synth such as an ARP Odyssey or ARP 2600 you will find lots of 'metallic' sounds. This is because they have a 'ring modulator' — a circuit that produces the dense cluster of non-harmonic overtones that are characteristic of metal sheets. With suitably chosen filters and envelopes, these can provide startling imitations that are, once again, quite beyond the capabilities of a simple harmonic oscillator.

> "A perfect 'noise generator' produces all audio frequencies simultaneously, and this is close enough to provide a basis for many analogue 'drum' sounds that would be quite unobtainable using conventional waveforms"

So there it is... Armed with a handful of conventional 'waveform' oscillators, a noise generator and a ring modulator, we're in a position to recreate the basic structures of almost all the most common musical sounds found in the 'real' (ie. non-electronic) world. Next month we will start to look at some of these, and show how we can use a simple subtractive synthesizer to recreate them.

Until then, have fun, and keep twiddling! SOS

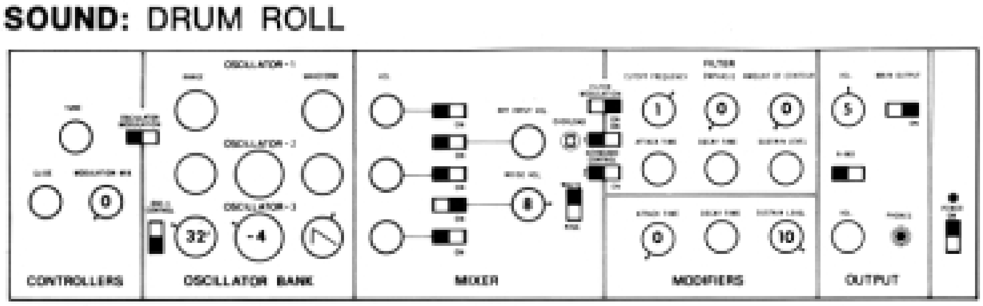


Figure 7: A 'drum' patch from the Minimoog's original "Sound Charts". Note that the conventional oscillators are all "OFF", with only the Noise Vol switch set to "ON" in the MIXER section.

# synth secrets

## PART 3: SIGNALS, MODIFIERS & CONTROLLERS

Gordon Reid moves on from discussing the harmonic components of sound to explaining how they change over time, and some of the tools subtractive synths give you to emulate this process.

If you have stuck with me for the past two months, you'll know that most natural sounds have complex harmonic structures, and that these derive from the nature of the object making the sound. You'll also appreciate that, no matter how you play them or what you do when you record them, percussion instruments will have significantly different tones from strings, pipes, and other 'conventional' instruments. But this knowledge is far from sufficient to allow you to understand and create the sounds you want artificially, with a synthesizer; you also need to know about the controllers and modifiers that shape the sounds you hear. Look at it this way... if you could define a sound purely by listing all the harmonics it was composed of, manufacturers wouldn't waste money putting all those 'unnecessary' filters and envelope generators and stuff in their products. So let's move on, and find out about how you can tailor a series of synth-derived oscillations into something more musical.

### Modifying A Sound

There are no sounds that you can define purely in terms of their harmonic spectra. Even if a sound seems to exhibit a consistent tone and volume, there must have been a moment when it began, and a moment when it will end. This implies that the loudness of the sound is *contoured* in some fashion.

Ignoring the start and finish for a moment, unvarying tones are called *static* or *stationary* sounds, and they are almost always musically uninteresting. Since no natural sounds are stationary over any significant timescale, the only time you are likely to encounter them is when they have been created by a *signal generator* (such as those found in analogue synthesizers, for example). You can think of these as devices that generate a tone by outputting a fluctuating voltage that is passed through an amplifier and then onwards to a speaker, which converts the voltage into sound that you can hear (see Figure 1).

Let's try to make this idea a bit more meaningful. Imagine that the amplifier in Figure 1 is your hi-fi amp, and that — although the tone generator is generating a signal — the volume knob is turned fully anticlockwise so that you hear no sound. Imagine you now turn the control fully clockwise while the sound is playing, and then turn it back anticlockwise again until silence returns.

If you look at Figure 2, you will see that you have added a controller (your manipulation of the knob) that causes the amplifier to modify the audio signal presented to it. But twisting a knob every time you want a non-static sound is hardly sensible, nor are the results precisely reproducible. Moreover, it's quite inappropriate if you are looking to use your tone generator to produce conventional sounds and notes. So you need to replace your fingers with a controller signal that gives predictable, reproducible results each time it is used. And it is this that brings us to the important concept of Voltage Control.

Imagine that the Controller in Figure 2 is another fluctuating voltage of some form. This is called a Control Voltage, or CV. Don't worry, for the moment, about how it is generated; just consider that, for any given voltage applied at the amplifier's 'Controller' input, the Amplifier applies a defined Gain to the signal. This describes a Voltage Controlled Amplifier, or VCA. Now, let's see what you can use to generate these CVs.

### Envelopes

Let's return to the idea of modifying a sound using the volume knob on the front of a hi-fi amplifier. Let's say that, if the knob is turned fully anticlockwise, the applied CV is 0 Volts, and the amplifier's Gain is zero. In other words, silence reigns. At the other extreme, let's say that, if the knob is rotated fully clockwise, the CV is 10V, and the Gain is maximum — ie. the sound is at its loudest. You could then imagine using the knob to apply varying voltages to create a loudness 'contour' for our sound. For example, the CV could start at 0 Volts, rise to 10V, drop to 5V, linger there for a while, before returning to 0V some time later. This profile is shown in Figure 3 on page 192.

As you can see, the contour of the CV is identical to the loudness contour. In other words,

▶

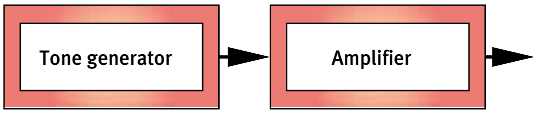


Figure 1: A simple sound generator.

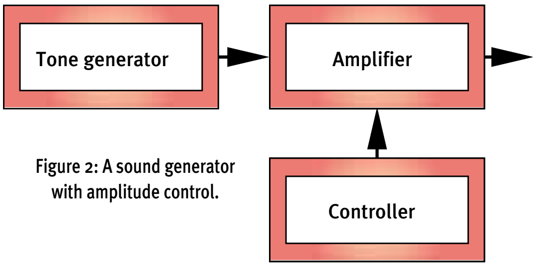


Figure 2: A sound generator with amplitude control.

you have defined the loudness of the sound at any given time using the CV. The shapes in Figure 3 are called Envelopes, so one of the devices that you can use to control our amplifier is, not surprisingly, called an Envelope Generator. EGs may be basic or complex, but (if they are not themselves modified in some way by another signal) they all share one attribute: each time they are initiated (or 'triggered') they provide a consistent contour, both in terms of the CVs produced and the times taken for them to change.

The most famous, and for a long time the most common envelope generators, are called ADSRs. The acronym stands for Attack/Decay/Sustain/Release, and these names represent the four stages of the EG. Three of them — Attack, Decay, and Release — are measures of time, while the fourth — the Sustain — is a voltage level. (See Figure 4.) The ADSR is, in many ways, a stroke of genius and, despite its great simplicity, it provides approximations to the contours of the sounds generated by a huge number of natural instruments.

For example, imagine the sounds produced by things as different as a pipe organ, a trombone and a thunderclap. Now consider how the loudness contour of each of these can be described in terms of its A, D, S and R stages. Remember:

- The Attack time determines the speed at which the sound reaches its maximum loudness.
- The Decay time determines the speed at which the loudness drops until it reaches...
- ...the Sustain Level, the level the loudness maintains until...
- ...it decays to its final level (usually silence) in a time determined by the Release time.

The organ has a rapid attack and maintains its full volume before dropping to silence when the player releases the key. Hence its loudness contour is almost precisely rectangular. Indeed, this shape has become so closely associated with organs that it is often referred to as an 'organ envelope', even when it is used in sounds that bear no relation to organ itself.

By contrast, the trombone 'speaks' more slowly, and its loudness usually peaks at the end of the attack stage before falling back to a lower, sustained level. When the player stops blowing, the sound rapidly falls back to silence.

Quiet different from either of these, the loudness of a thunderclap often develops relatively slowly, and there are no decay or sustain stages; once it has peaked, the loudness dies away slowly.

As you can see from Figure 5, these contours are fundamentally different from one another. Let's take the trombone envelope first. This requires all four ADSR parameters, with moderate attack, decay and release times and a moderate sustain level. The organ envelope is clearly simpler, and requires just three of the ADSR envelope's parameters, with virtually instantaneous attack and release, and a maximum sustain level. In contrast, the thunderclap only utilises two parameters. It has no Sustain or Release values (actually, this isn't always true, but we'll discuss

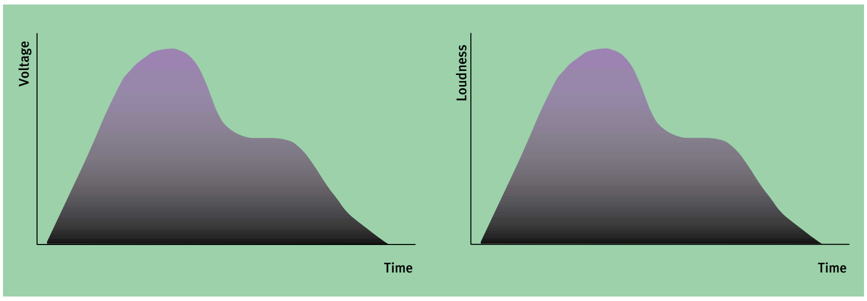


Figure 3: A voltage control becomes a loudness contour.

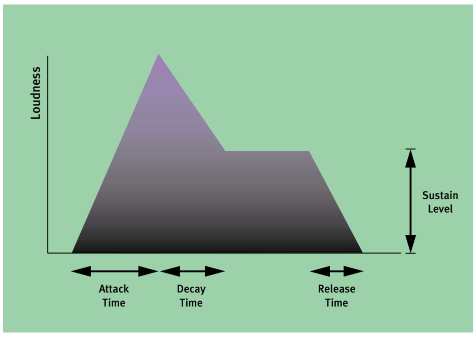


Figure 4: The ADSR Envelope.

"The ADSR is, in many ways, a stroke of genius and, despite its great simplicity, it provides approximations to the contours of the sounds generated by a huge number of natural instruments."

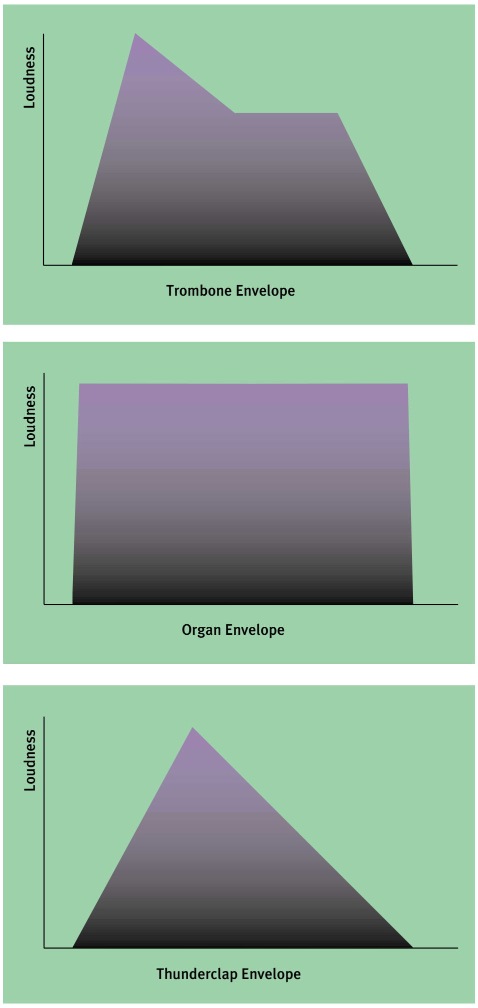


Figure 5: A selection of ADSR contours.

that in a later part of this series).

However you set the parameters of the envelope generator in your synth, if it is connected to the VCA, it has simply replaced the general term 'controller' in Figure 2. Provided that you can trigger it at will, you have a device that will shape this aspect of your sound whenever you want.

## Low Frequency Oscillators And Tremolo

Let's return for a moment to the concepts relating to oscillators, described in Part 1 of this series (see *SOS* May '99). You will remember that every harmonic sound has a fundamental frequency that is the simplest mode of vibration of the oscillator producing it. If this fundamental lies in the range of approximately 20Hz (20 vibrations per second) to 20kHz (20,000 vibrations per second) you should hear the sound as an audible tone.

Now consider again the hi-fi amplifier and its volume knob. If you swing the control from side to side once or twice per second you will introduce a new, periodic effect to the sound: that of tremolo. You are, in essence, applying an oscillator to the hi-fi's volume. And although the frequency of this oscillator is much less than 20Hz, its effect is clearly musically important.

It should come as no surprise, therefore, to find that most synthesizers have dedicated devices — Low-Frequency Oscillators (LFOs) — generating low-frequency signals that control many of the synth's other functions. On most instruments, the LFO(s) produce oscillations in a range of frequencies lying between about 0.1Hz (one cycle every ten seconds) and 20Hz. These are suitable for producing relatively simple sonic effects, and as you can see in Figure 7, tremolo is just a more specific example of the setup shown Figure 2: in this case, it's an LFO rather than an EG controlling the gain of the amplifier.

Trivial though Figure 7 may be to most readers, it also demonstrates the three types of modules present in all synthesizers. In this configuration:

- The Tone Generator is a *Signal Generator* — it produces the basic audio tone;
- The Voltage Controlled Amplifier is an example of a *Modifier* — it changes the audio signal in some way;
- The LFO Sinewave generator is acting as a *Controller* — it is directing the signal-modifying action of the Modifier itself.

But simple as this example is, you can use this architecture to produce some extremely complex sounds. You just need to change one detail in the setup...

The LFOs on more powerful synths are often capable of producing higher-frequency oscillations that stray well into the audio range. Furthermore, these LFOs often offer a wide range of waveforms. But if you can modulate a signal with an audio-frequency LFO, why can't you use another Tone Generator to do so? Of course, there's no reason why you shouldn't, and the architecture in Figure 8 allows you to create the complex sounds alluded to above.

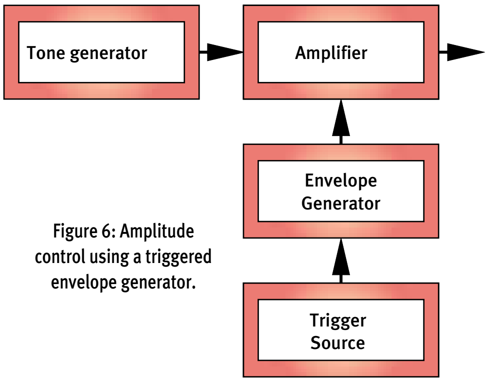


Figure 6: Amplitude control using a triggered envelope generator.

## ... And The Point Is ...?

Although this article has never strayed from the basics, three major concepts have been introduced: Control Voltages, Envelope Generators, and Low Frequency Oscillators. But these are not the most important lessons to learn.

Look again at the diagrams in this article. In each of these the audio signal has been represented by the horizontal arrows, while controlling signals have been shown as vertical arrows. I like to think of these as (i) signals that you hear, and (ii) signals that control what you hear. Obvious, huh? But on the other hand, you must also recognise that in voltage terms, there is actually no difference between these signals. Consequently, many synth modules can act equally as signal generators, modifiers, and controllers, depending only upon where they are placed (and how they are used) within the sound-generating architecture.

In other words: an analogue synth uses fluctuating voltage to represent audio signals and other fluctuating voltages to shape and control them. But it's not a signal's source that is important, it's the destination that determines whether it is best viewed as an audio signal or a controller.

And that is one of the most important synth secrets of them all. SOS

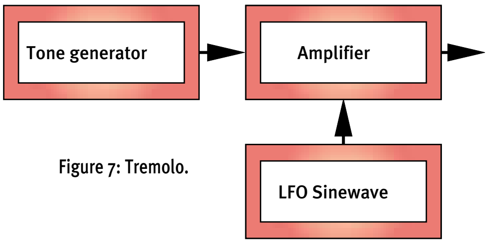


Figure 7: Tremolo.

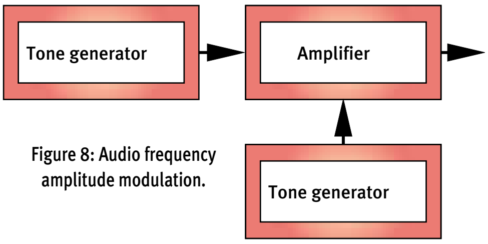


Figure 8: Audio frequency amplitude modulation.

"...it's not a signal's source that is important, it's the destination that determines whether it is best viewed as an audio signal or a controller."

# synth secrets

## PART 4: OF FILTERS & PHASE RELATIONSHIPS

Having dealt last month with the concepts of envelopes, oscillators and LFOs, **Gordon Reid** moves on to the subject of filters, and the havoc they wreak on the signals that pass through them.

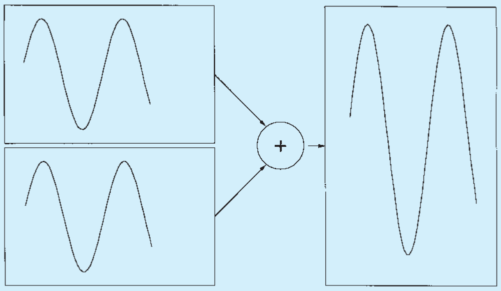


Figure 1: Perfect addition of in-phase sine waves.

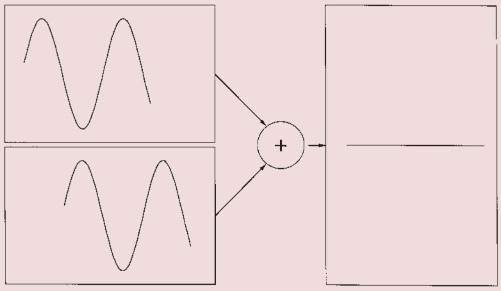


Figure 2: Perfect cancellation of two sine waves offset by half a cycle.

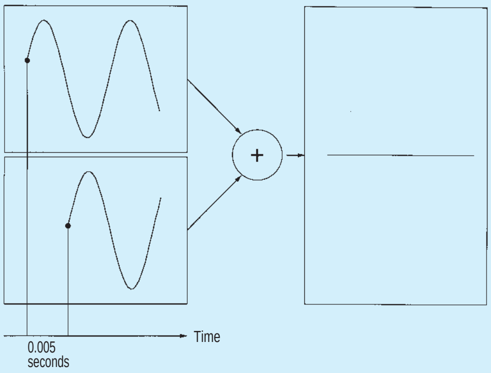


Figure 3: Perfect cancellation of a 100Hz sine wave shown as a timing difference.

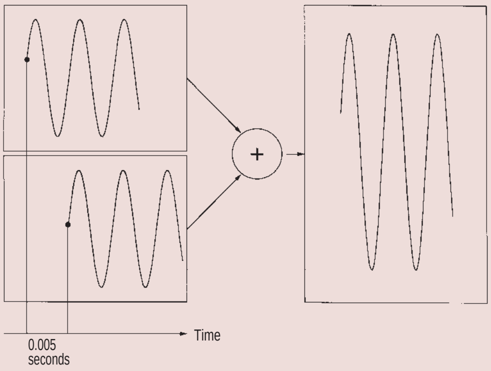


Figure 4: Perfect addition of a 200Hz sine wave offset by the same amount.

So, you've read parts 1, 2 and 3 of this series, and you've digested everything I've written so far about oscillators, envelope generators, VCAs, and LFOs, how they work, what they do, and how they slot into the typical subtractive synthesizer. About now, you imagine, it ought to be time to get the low-down on the crowning glory of subtractive synthesis — the filters. Well, yes... and no. If you're expecting something like "wind the resonance up to 11, and sweep the 24dB/octave VCF for some classic tearing analogue sounds," you're going to be disappointed.

If you really want to know why some filters sound good and others are so uninspiring, you've got to understand more about them. Unfortunately, filters are the most misunderstood parts of any synthesizer, and even the most basic belief surrounding them (that they just make parts of the signal quieter) is wrong. A filter — analogue or digital — does far more than just attenuate... it really screws with your signal.

To see exactly what happens when you pass a signal through a filter, it's necessary to know something about phase relationships. And to hack that, you need to take another look at the sine waves introduced in the first part of this series.

### Just A Phase

Let's start by considering what happens when you combine two sine waves using a simple mixer. As you might imagine and as you can see in Figure 1 (right, top), adding together two identical waves produces the same sound, but louder. But what happens if you start the lower wave halfway through the cycle of the upper one? Figure 2 (below Figure 1) shows that, if you add these waves together, they cancel each other out, and you hear nothing. Although, in isolation, these signals would sound identical, combining them results in silence.

This is an important result, which demonstrates that while you can describe a single sine wave by specifying just its frequency and amplitude, if you combine two or more waves you must consider their relative offset. This offset is usually called the 'phase' of one wave with respect to the other, and is expressed in degrees, like the ones used to describe angles (if you want to know why, have a look at the 'Expressing Phase In Degrees' box below, but if you can't be bothered with another technical explanation at this point, read on).

Of course, you can mix sine waves with any offset (ie. any phase difference) and the amplitude of the result will lie somewhere between the 'double' loudness of Figure 1 and the silence of Figure 2 (if, instead of combining the signals in mono, you play the waves independently through stereo speakers, you'll get a quite different result. But that's a topic for another day, and we won't discuss it further here).

Instead, let's consider this offset as a timing difference. Let's say that our sine waves have a frequency of 100Hz or, to use older terminology, oscillate at a frequency of 100 cycles per second. We can say, therefore, that each cycle takes 0.01 seconds and, for these signals, an offset of half a cycle (a phase shift of 180º, if you've read that box yet, and if you haven't, don't worry) is equivalent to a time shift of 0.005 seconds or, as it is more commonly expressed, 5 milliseconds. Figure 3 on page 116 should help to make this clearer.

But now consider a different pair of sine waves at twice the frequency of the first. These have a frequency of 200Hz, and so five milliseconds is now sufficient time to complete a *full* cycle. In this case, as shown in Figure 4 on page 116, our two

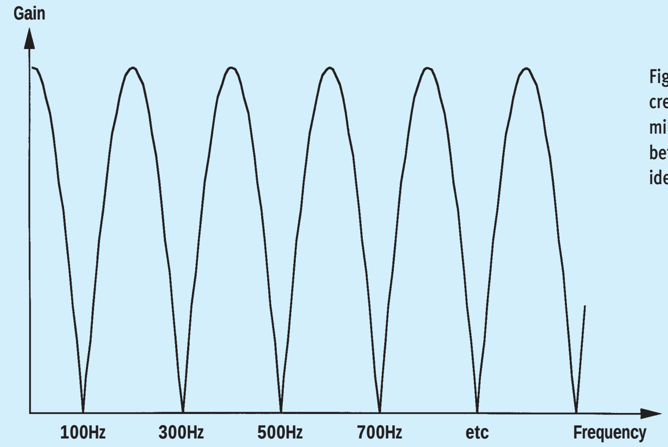


Figure 5: The filter created by a 5 millisecond offset between otherwise identical signals.

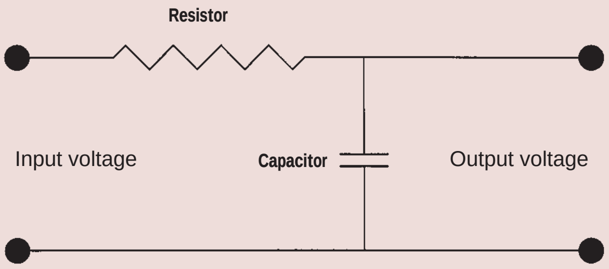


Figure 6: A simple 'RC' low-pass filter.

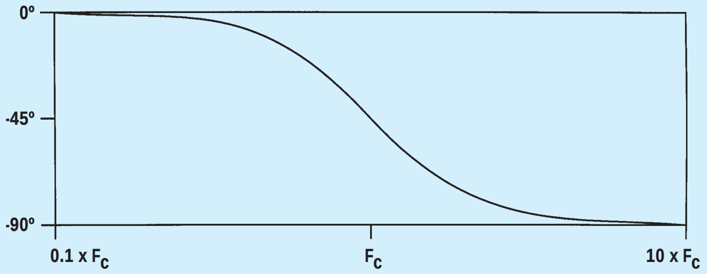


Figure 7: The phase response of our RC filter.

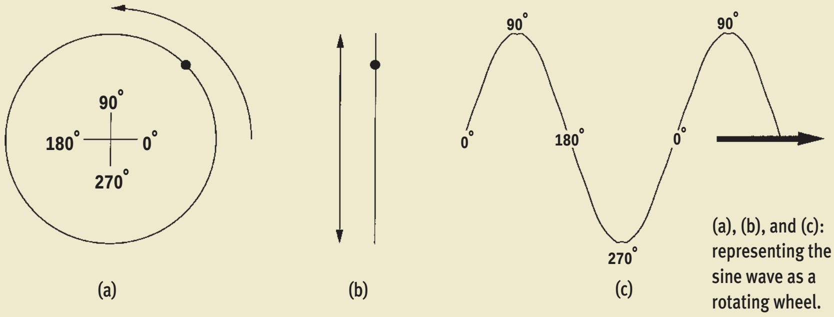
## Expressing Phase In Degrees

**How is it that we can express phase shifts in degrees? After all, degrees are used in maths to measure angles, or amounts of rotation, aren't they? Well, it all depends on how you look at things. Consider a simple sine wave (see (c) below right). At some arbitrary time, the wave starts rising from zero, peaks exactly a quarter of the way through its cycle, crosses the zero line again halfway through, reaches its lowest point three-quarters of the way through the cycle, and then returns to zero before starting all over again.**

**The shape of the sine wave can, however, be described in a different way. Imagine a spot on the circumference of a wheel rotating at a constant speed (as shown in (a) on the right) — and imagine the shape it traces if you only consider the vertical element of its movement (this may seem like a rather arbitary thing to do, but trust me on this a bit longer). You'll get an up-and-down trace like the one shown in (b), right. Finally, imagine drawing that up-and-down motion on a piece of paper moving past it at a constant speed. Can you see that the result (shown in (c) here) would be a sine wave again? If not, blame my drawing skills (or lack thereof), because it is.**

**Working backwards, this approach allows us to employ the same units used to describe the rotation through which the wheel passes — degrees — to describe the progression through the sine wave cycle. The sine wave starts to be traced out when the spot on the circumference of the wheel is at the position that convention calls 0˚. At the top of the cycle, the spot has rotated through 90˚. It drops past zero again at 180˚, has its lowest point at 270˚, and returns to the start after rotating a full 360˚ (which is the same thing as being at 0˚ again).**

**This is a very neat way of expressing this physical attribute of the waveform, and makes it easy to describe relative phase. For example, if two sine waves are offset by half a cycle with respect to each other, you can say that (because one is at 180˚ when the other is at 0˚) these waves are '180˚ out of phase'.**



(a), (b), and (c): representing the sine wave as a rotating wheel.

▶ sine waves *add* if one is delayed by 0.005 seconds. This is because two signals that are offset by a complete cycle (360º out of phase) are once again perfectly in phase with each other!

Right, let's take a deep breath, and apply these ideas to a more complex waveform — say a sawtooth. As you may remember from the first part of this series, a sawtooth wave has every harmonic present, so, if the fundamental (the first harmonic) lies at 100Hz, the second harmonic will be at 200Hz, the third at 300Hz... and so on. Adding two of these sawtooths (sawteeth?) with the fundamentals offset half a cycle means, of course, that the fundamentals are cancelled out. But the second harmonics, lying at 200Hz, will be added! The third harmonic, at three times the frequency of the fundamental, will be cancelled out, the fourth harmonic will be reinforced, the fifth cancelled... and so on. The result is a waveform with harmonics at 200Hz, 400Hz, 600Hz and so on... in fact, it's a sawtooth of the same amplitude but exactly twice the frequency of the originals.

This is a stunning result, quite at odds with much perceived wisdom. So this month's first Synth Secret is:

*Combining complex 'out of phase' signals does not necessarily lead to complete cancellation. In fact, in the real world, it rarely,* if ever*, does so.*

Yet even this result is one of the simplest cases of phase-shifting you can consider. Imagine how much more complex the result becomes when you apply these ideas to a complex waveform: some harmonics will be a bit quieter, some will be a bit louder, a few will be completely cancelled, and a few will be perfectly added. But this is, of course, what happens in the real world. Fourier analysis (back to part one again, for anyone who can't remember) tells us that any two complex signals — such as those representing speech or music — can be described as an infinite number of sine waves that represent all the frequencies present in the signal. So, for any given offset between two otherwise identical signals, each frequency will be phase-shifted by a different amount. The result, when viewed on a spectral analyser, looks like a broad comb, with the distance between the 'teeth' of the comb (the cancellations) defined by the timing difference (see Figure 5, on page 118).

In other words, when you mix two offset but otherwise identical signals, the phases of the individual frequencies define a filter. This, because of its characteristic shape, is called a Comb Filter, and you'll find examples of this on instruments as diverse as the Analogue Systems RS Integrator (a traditional modular analogue synth) and the DSP-based Waldorf Q.

## Phasing & Filtering

OK, so now you know something about phase, and you can see that this can be closely associated with audio filtering. But ask yourself this: If phase changes lead to filtering, can you assume that filtering leads to phase changes? The answer is, of course, yes.

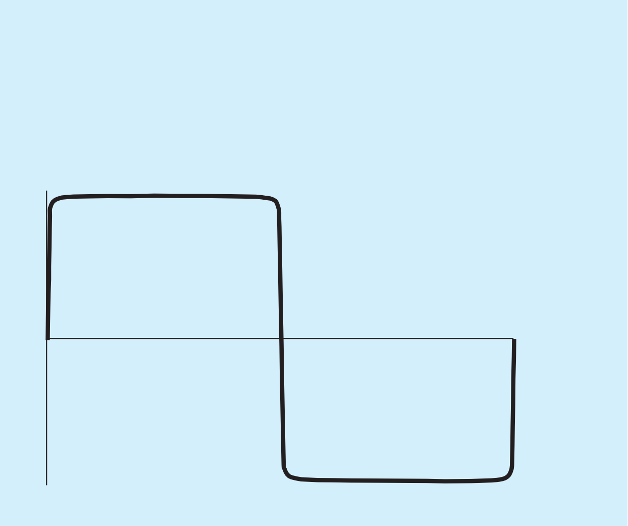


Figure 8: An ideal 100Hz square wave with all harmonics up to 20,000Hz included.

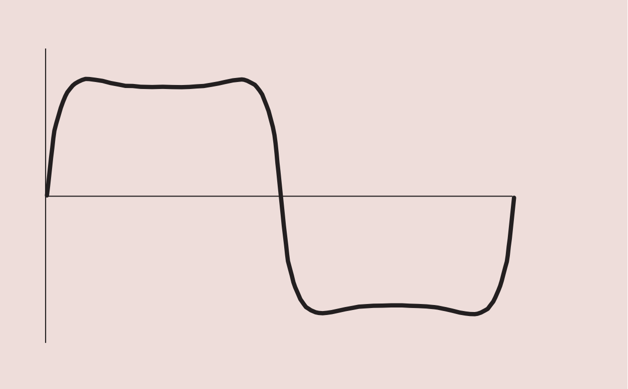


Figure 9: The idealised waveform produced by filtering a 100Hz square wave using our RC filter with the filter cutoff (Fc) equal to 400Hz.

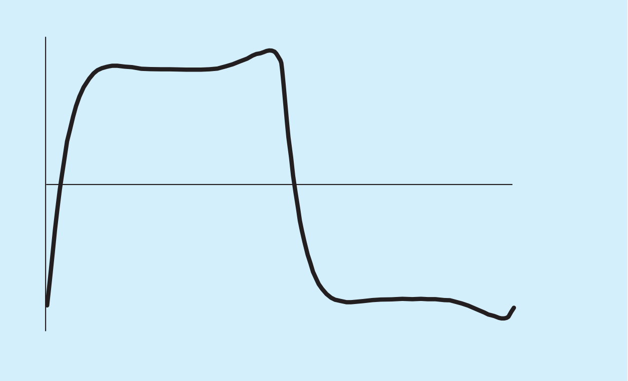


Figure 10: A calculation of the true waveform produced by filtering a 100Hz square wave using our RC filter with the filter cutoff (Fc) equal to 400Hz.

Look at the electrical circuit in Figure 6 (on page 116). This has just two components — a resistor and a capacitor — but it depicts a perfectly usable filter called an RC low-pass filter. And, as any novice synthesist knows, a low-pass filter passes unhindered all the frequencies below a 'cutoff' frequency while attenuating all those above it. For this simple filter, the cutoff frequency is defined by the component values, while the nature of the circuit itself defines the rate at which the higher frequencies are attenuated.

Surprisingly (you may think) we're not going to worry about the rate of attenuation this month — we'll cover that next month. Instead, we're going to look at what this filter does to the phases of the signals fed into it...

Look at Figure 7 (on page 116). This describes a property called the 'phase response' of our simple LPF, and it shows us that the phase of any given frequency presented at the filter's input will be shifted backwards to a greater or lesser extent. As you can see, low-frequency signal components are largely unaffected by the filter, a component at the cutoff frequency is shifted by exactly an eighth

**Many thanks to Dave Betts and Christopher Hicks of CEDAR Audio, who have tried to ensure that I don't end up with signal-processing egg on my face.**

of a cycle ( or -45º), and high-frequency components are shifted by a full -90º.

Since these concepts are a bit esoteric (and because I want you to read to the end of this article without getting a headache) let's illustrate this by seeing what our RC filter does to something as fundamental to an analogue synth as a 100Hz square wave.

If you recall the first part of Synth Secrets, you'll remember that you can represent any conventional waveform by a number of harmonics, called the fundamental and its overtones. In this case, our input signal (the square wave) has a fundamental of 100Hz. The second harmonic — at 200Hz — is absent, but the third harmonic lies at 300Hz and has an amplitude of a third of the fundamental. Similarly, the fourth harmonic is absent, but the fifth is present at 500Hz and has an amplitude of one-fifth... and so on. All the harmonics are in phase, and the waveform looks like that shown in Figure 8, left.

Now let's say that our simple RC filter has a cutoff frequency of 400Hz, and imagine what would happen to our square wave if the filter's phase response was zero at all frequencies. This is quite simple: the fundamental and first overtone of the square wave (the harmonics at 100Hz and 300Hz) would be unattenuated, but all the overtones at 500Hz and above would be attenuated according to the filter's response. The resulting waveform (and you'll have to trust me on this) is shown in Figure 9, left.

But now let's take into account the phase shifts imposed upon each of the harmonics in the signal. We now get a waveform that looks very different, and the true output from our filter (Figure 10) is visibly distorted when compared to the original.

This leads to a hugely important conclusion, and this month's most important Synth Secret:

*Filters not only change a waveform by attenuation, but distort it by individually phase-shifting the harmonics within it.*

Strangely, due to the relative simplicity of the filtered square wave, you probably won't be able to hear the difference between the waveforms in Figures 9 and 10 — you would need a more complex wave to hear the phase-shifting of the harmonics. But as you've already seen in this series, there are very few sounds in the real world that *aren't* more complex than a square wave, and so the effect on most sounds can be quite dramatic. And, of course, if it's a *Moog* filter you're passing the sound through... ah, but that's a discussion for another day.

**"...if phase changes lead to filtering, can you assume that filtering leads to phase changes? The answer is, of course, yes."**

# synth secrets

## PART 5: FURTHER WITH FILTERS

Gordon Reid continues his series on the theory of subtractive synthesis by delving deeper into the amazingly complex world of the analogue audio filter.

Last month, we started looking at audio filters, demonstrating (if not proving) that they are also phase-shifters, and therefore mess with your audio signals in more ways than one. In doing so, we skipped over the most important aspect of such devices: their abilities to remove parts of the audio spectrum and create new tones from old. But if you think that this is the easy bit, or that you already know how filters attenuate signals, the chances are that you're wrong (on both counts). So let's get stuck in, and dispel a few myths...

### Passive Filters

Figure 1 (see right) shows the passive low-pass RC filter first introduced last month (incidentally, you can, with a fair degree of accuracy, define a passive component as one that draws no power except that presented as the signal at its input — so resistors, capacitors and inductors are examples of passive components, while transistors and other amplifiers are not). If you read last month's instalment, you may remember that we can define the cutoff frequency of an RC filter simply by choosing appropriate values for the two passive components within it. So far, so good... that tells us where in the audible spectrum the filter will take effect, but is doesn't tell us what the extent of its effect will be.

The relationship between what you put into a filter and what you get out is called the Transfer Function, and strictly speaking, this should encompass both the amplitude response (ie. effect on volume) and the phase response of the filter. But, since we discussed the phase-shifting aspect last month, we're only going to consider the amplitude response this month. As it happens, the idealised transfer function of our RC filter is very simple: for every doubling of the frequency above

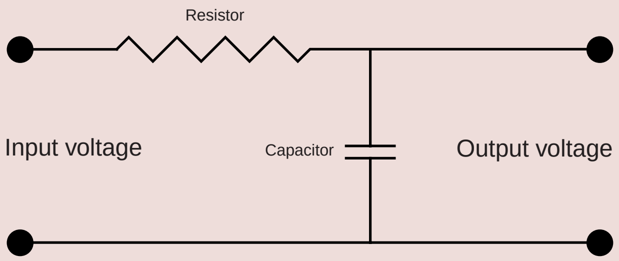


Figure 1: A simple low-pass filter.

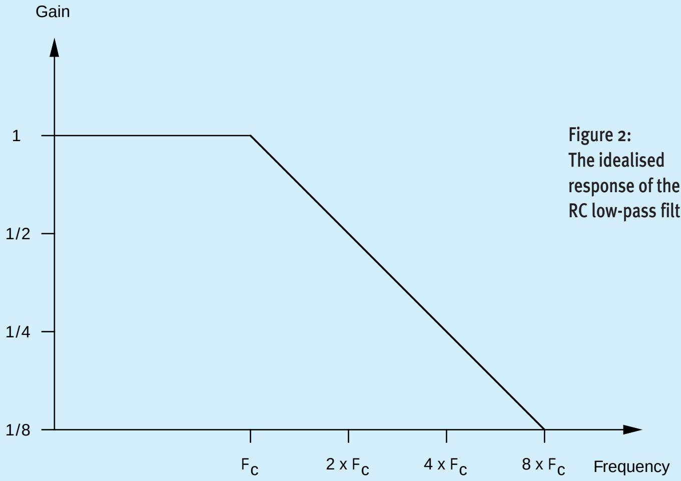


Figure 2: The idealised response of the RC low-pass filter.

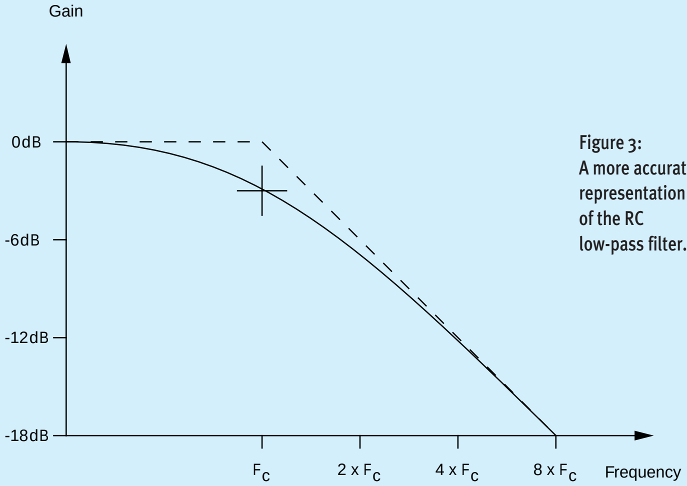


Figure 3: A more accurate representation of the RC low-pass filter.

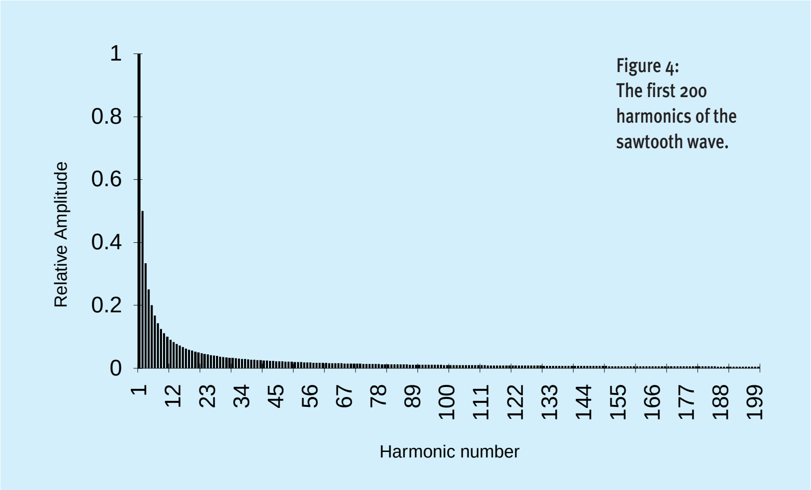


Figure 4: The first 200 harmonics of the sawtooth wave.

the cutoff frequency (which I will call Fc), the gain at the output is halved (see Figure 2, left).

So, for example, if Fc is 1kHz, the gain at 2kHz is 1/2 (ie. the output is halved), the gain at 4kHz is 1/4 (the output is a quarter)... and so on. Since each doubling of the frequency is equivalent to moving up an octave, and each successive halving of the gain is known as an attenuation of six decibels (6dB), this response is most commonly called a 6dB/octave filter.

Unfortunately (and despite its ubiquitous use within the music industry) Figure 2 is actually *wrong*. Figure 3 shows a more accurate representation of the transfer function. As you can see, the signal amplitude is already down by 3dB at the cutoff frequency. This is not a fault. In fact, in electrical engineering, the position of this 3dB cut *defines* the cutoff frequency. So, let's state this month's first Synth Secret:

*The cutoff frequency of a passive low-pass filter does not define the frequency at which the filter starts to work; it is itself defined as that frequency at which the signal is already attenuated by 3dB. And, since an attenuation of 3dB is easily perceived by the human ear, this means that you are already significantly affecting the signal at the cutoff frequency.*

Now, let's back-track a little and consider what a simple low-pass filter does to a common waveform. To simplify matters, we'll use the idealised low-pass filter response seen in Figure 2, because its sharp 'knee' makes it easier to recognise what's happening. Figure 4 shows the harmonic structure of the most common analogue synthesizer waveform of all: the sawtooth. All the harmonics are present in this signal, and their ▶

amplitudes relative to the fundamental are defined by the simple relationship 1/(harmonic number). Drawn on conventional graph axes, the amplitudes of the first 200 harmonics look like the graph shown in Figure 4 (see page 101). However, Figure 4 is far from the best way to represent these harmonics. Much better is the graph with logarithmic axes shown in figure 5 (right). This looks quite different, but it represents exactly the same information, so don't worry if you don't know what a logarithmic scale is. Furthermore, it should be obvious why I have chosen to change the axes of the graph in this way, even if you *don't* know what a logarithmic scale is: unlike in Figure 4, the amplitude relationship is now a straight line, and this makes it easy to see the filters' effects in the following graphs. Indeed, if you now look back at Figures 2 and 3 and study the axes, you will see that these graphs also have logarithmic axes.

## Applying The Filter

So let's see what a 6dB/octave RC filter with a cutoff frequency of, say, 3kHz does to the harmonics and waveform of a 100Hz sawtooth wave. Figure 6 (right) shows how the filter attenuates the frequencies above 3kHz. If you bother to measure the new slope, you'll find that the additional 'rolloff' (as the attenuation is commonly known, for reasons which should be obvious from looking at the graph) conforms to the 6dB/octave rule mentioned above. If you now look at Figures 7 and 8 (see right), you'll see that the first diagram shows our idealised 100Hz sawtooth waveform with all its harmonics up to 20kHz unattenuated, while the latter shows the same signal processed by our 3kHz filter. As you can see, there's not much visible difference between the two waveforms. This is because the 3kHz cutoff frequency allows the first 30 harmonics through untouched, and it's only the low-amplitude high-frequency harmonics that are affected. Nevertheless, the human ear's enormous sensitivity ensures that you hear even this tiny difference as a 'dullness' or lack of 'top end' in the filtered sound.

> "Filters with a 6dB/octave characteristic are used as tone controls in stereo systems, and occasionally within synthesizers as supplementary brightness controls, but they are not much use for true synthesis."

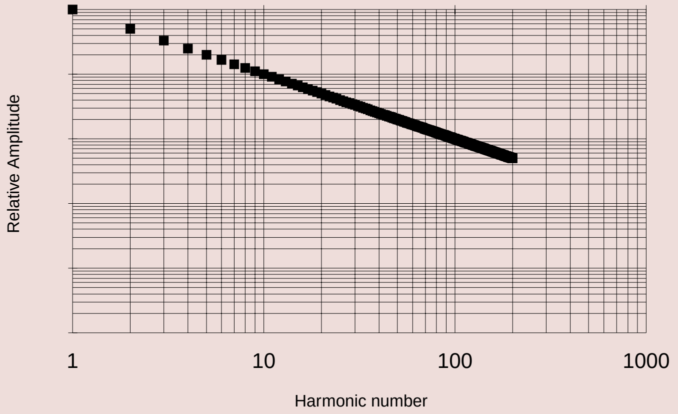


Figure 5: The harmonic structure of a 100Hz sawtooth wave expressed on log/log axes.

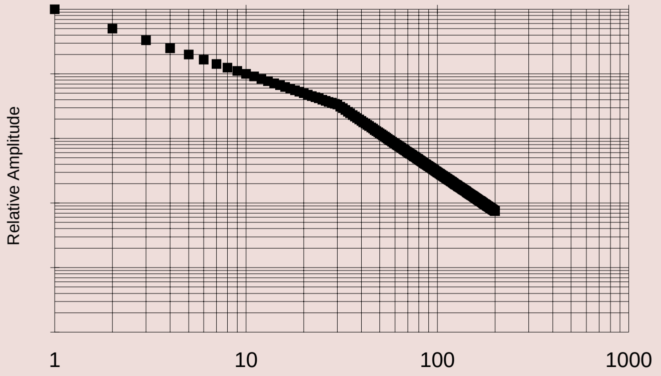


Figure 6: The action of a 3kHz, 6dB/octave low-pass filter applied to a 100Hz sawtooth wave.

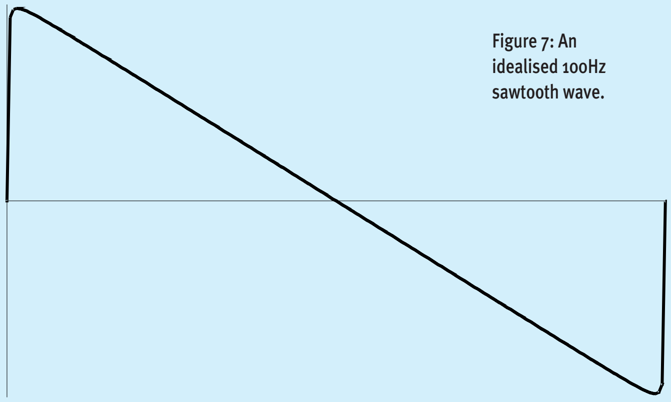


Figure 7: An idealised 100Hz sawtooth wave.

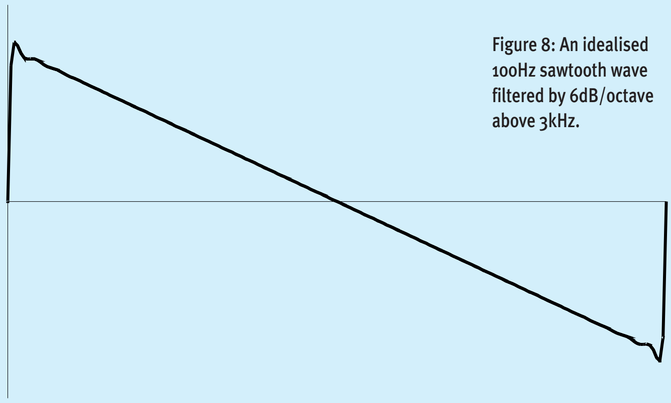


Figure 8: An idealised 100Hz sawtooth wave filtered by 6dB/octave above 3kHz.

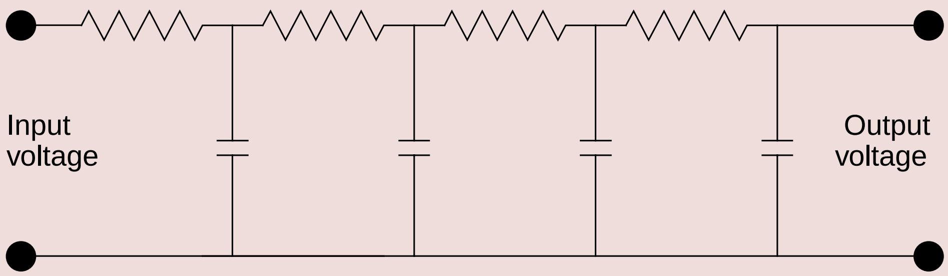


Figure 9: A cascaded 24dB/octave low-pass RC filter.

## Steepening The Slope

Filters with a 6dB/octave characteristic are used as tone controls in stereo systems, and occasionally within synthesizers as supplementary brightness controls, but they are not much use for true synthesis. This is because they don't modify the waveform enough to change the tone very dramatically — filtered signals just sound like the original, only duller. Clearly, a more powerful filter is required if new timbres are to be created.

So what passive components allow us to create simple circuits with more powerful 12dB/octave, 18dB/octave or even 24dB/octave attenuations? Unfortunately, none of them do — so a different approach is needed. Why not cascade a number of RC filters to create the required steeper rolloffs? For example, two filters could be used together to create a 12dB/octave filter, three for an 18dB/octave filter, and four for a 24dB/octave filter. The four-element filter would then look like the idealised circuit shown in Figure 9 (above), and you would expect its idealised transfer function to look like Figure 10.

However, sadly, it doesn't work as simply as that. Our model of the response of the passive RC filter requires certain assumptions about the inputs and outputs; and while you can just about satisfy these assumptions for a single RC circuit, they break down completely if you try cascading elements as I've just proposed. So where do we go from here?

## Tricks Of The Trade

**There are a number of practical tricks that you can employ to overcome the inherent deficiencies of audio filters. For example, when designing his Fénix semi-modular (reviewed in *SOS* February '99) Marc Paping of Synton used a 5-pole design (which has a theoretical maximum rolloff of 30dB / octave) to ensure that the Fénix's 'Moog-style' low-pass filter actually achieved a true 24dB / octave rolloff in the audio band.**

**On the other hand, it's also easy to get it wrong. A simple calculation error led to the infamous ARP4075 filter used in some ARP synthesizers. In theory this was a first-class filter, but the cutoff frequency wouldn't exceed 12kHz or so, and this made the instruments sound dull and lifeless when compared to their predecessors. Amazingly, you could easily correct the problem by changing just four resistors on the circuit board!**

## Up the Pole

If you've been in the synth game for a while, you'll have heard that 12dB/octave filters are sometimes called '2-pole' filters, and 24dB/octave filters are called '4-pole' filters. You might think it safe to assume, therefore, that each of the 6dB/octave sections in Figure 9 is a 'pole'. Unfortunately, you would be wrong (although not a million miles from the truth).

The name (in this context) is a consequence of a powerful mathematical operation called a 'Laplace Transform'. This transform, while difficult to describe in words, is a convenient operation that allows mathematicians to analyse the responses of linear systems when they are presented with audio signals (as for 'linear systems' and the maths involved... no, don't even dream of asking!) Anyway, the term 'pole' comes about because, when you represent an RC filter using a graph in the 'Laplace Transform domain', it looks like a flat sheet of rubber with a tent-pole pushing it sharply upwards at some point. A single 6dB/octave RC filter has one such 'tent-pole', and is therefore called a '1-pole' filter, a 12dB/octave filter has two 'poles'... and so on. Therefore, if you want to create a passive 24dB/octave filter with a single cutoff frequency for each of its four elements, it would seem safe to assume that would you want all the poles in the same place in the graph. And, for once, intuition is correct. Unfortunately, as I've already explained, achieving this using passive components is all but impossible because, when we cascade the sections, they interact and no longer function as they would in isolation. So, instead of the perfect 24dB/octave response of figure 10, the cutoff frequency for each section is different, and the amplitude response of our transfer function has four 'knees', as shown in Figure 11 (see right).

This then, leads us to an important conclusion:

> **"the term 'pole' comes about because, when you represent an RC filter using a graph in the 'Laplace Transform domain', it looks like a flat sheet of rubber with a tent-pole pushing it sharply upwards at some point."**

while a passive 4-pole filter will tend to a 24dB/octave rolloff at high frequencies, it will, to a greater or lesser extent, exhibit regions within which the rolloff is 6dB/octave, 12dB/octave and 18dB/octave. Moreover, if you look closely, you'll see that the transfer functions within these intermediate regions are not quite straight lines, meaning that the relationship between the frequency and the input and output powers are not as straightforward as before.

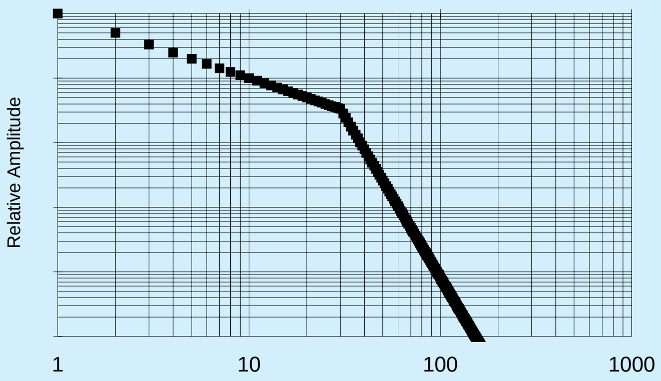


Figure 10: The hypothetical attenuation of high harmonics caused by a cascaded 24dB/octave low-pass filter.

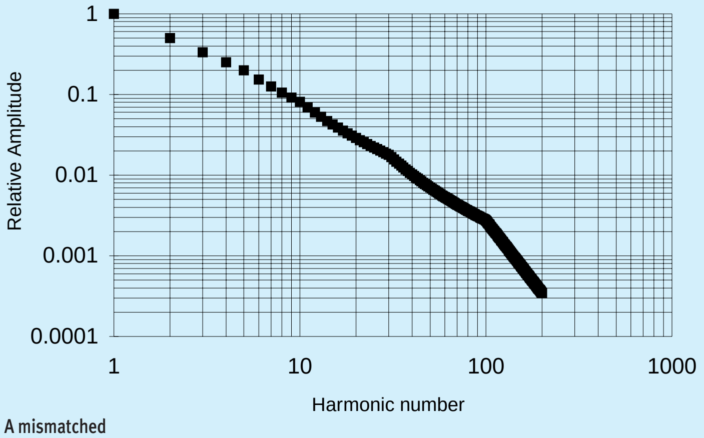


Figure 11: A mismatched 4-pole filter.

## Let's Get Active

Let's recap our position so far: we've designed a circuit with an theoretical rolloff of 24dB/octave but, because the filter elements interact, the cutoff frequencies of each are different. Furthermore, the 'knee' of each cutoff is rounded and (trust me on this) even if the sections' cutoff frequencies were identical, the composite 'knee' would get no 'sharper'. We've also ignored any effects on the signal in the supposedly untouched 'pass band' below the cutoff frequency (which we know actually suffers attenuation to an extent, and may well endure other side-effects too) and, finally, we've completely ignored the phase-shifting effects of each of the filter stages.

You might not unreasonably conclude at this point that this design approach is pretty worthless — but fortunately, we can deal with some of the above problems by inserting components between the filter stages to separate (or 'buffer') the responses of each from the other. These components include operational amplifiers (more commonly called 'op-amps') and it is these that make a filter 'active'.

Unsurprisingly, given what we've just learned, the filters in all analogue synthesizers are active (with the exceptions of a few brightness controls and basic equalisers). These filters are more complicated than their passive brethren but have the advantage that, by suitable choice of design, you can make them respond in desirable ways. For example, you can concentrate on obtaining a sharper knee at the cutoff frequency, or maximise the flatness in the pass-band, or engineer a tailored phase response. Unfortunately, you can't optimise all of these simultaneously, and so as in many other areas of recording equipment manufacture, good filter design is often a trade-off between desirable attributes.

But there's another consideration: even when we discuss the fundamental problems of these filters, we treat the electrical circuits themselves as if all the components within them were 'ideal', and responded in ideal ways. Yet many components are rated to within 1 percent, 2 percent... sometimes 10 percent of their quoted values. This means that two ostensibly identical circuits will often be different in subtle ways. So, even in the case of an active 24dB/octave filter, it's highly unlikely that (for example) all four 'poles' will be perfectly superimposed, one upon the other. This means that our conclusion regarding passive filters is itself a generally applicable Synth Secret:

*A 4-pole filter will always tend to a 24dB/octave rolloff at high frequencies, but it will exhibit regions within which the rolloff is 6dB/octave, 12dB/octave, and 18dB/octave.*

If all this leads us to just one conclusion, it's this: filters are conspicuously more complex than we are often led to believe. Indeed, if you were to accept the common, simplistic definitions of 24dB/octave filters you might expect (say) the Minimoog's and the later ARPs' filters to sound the same. This is clearly not the case. Similarly, you might think that the MS20's and the Oberheim SEM's 12dB/octave filters would be similar, and this is equally false. So this month's final Synth Secret is something that you all knew anyway:

*Whether the differences are subtle or glaringly obvious, every analogue filter design sounds different from every other.*

# synth secrets

## PART 6: OF RESPONSES AND RESONANCE

As parts 4 & 5 of Gordon Reid's series showed, even the simplest analogue filters mess with your sound in complicated ways. In this part, he considers what happens when you make the design more sophisticated...

If you have read the most recent two parts of this series you'll now know (at least in principle) how to construct a 24dB/octave filter and define its cutoff frequency. You will also be able to appreciate how that filter messes around with the phases of the harmonics within any signal you pass through it. So now we can start talking about winding up the resonance to 11, overdriving the input and creating some classic tearing analogue filter sweeps... Yes?

No! All the filters we've discussed so far have been static, acting just like tone controls, albeit rather powerful ones. No matter how you use them, you will eventually end up with something that sounds about as interesting as a documentary on last year's party political broadcasts. So let's introduce another handful of vitally important ideas, tie up a few loose ends, and bring ourselves to the point where we can, finally, get twiddling.

### Step 1: Varying The Cutoff Frequency

We'll start, as usual, with our simple 6dB/octave low-pass RC filter (see Figure 1). Let's consider what happens if we replace the resistor at the top of the diagram with a variable resistor, more commonly known as a potentiometer. We then get the circuit shown in Figure 2. Now, I have done my best to avoid any maths, but there's something you ought to know about this filter: the cutoff frequency is directly related to the amount of electrical resistance provided by the resistor, and the capacitance of the capacitor. The relationship is so simple that I can write it like this:

$$f_c = \frac{1}{2\pi RC}$$

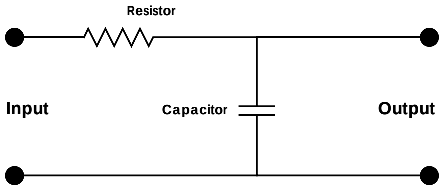


Figure 1: A passive 6dB/octave RC low-pass filter.

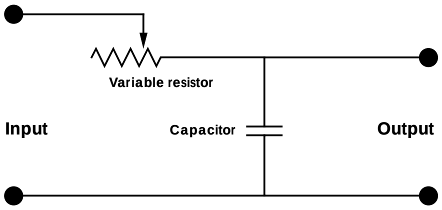


Figure 2: An RC low-pass filter with variable cutoff frequency.

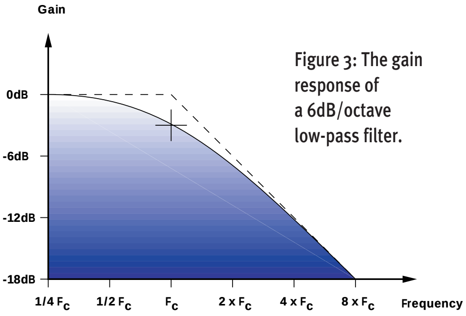


Figure 3: The gain response of a 6dB/octave low-pass filter.

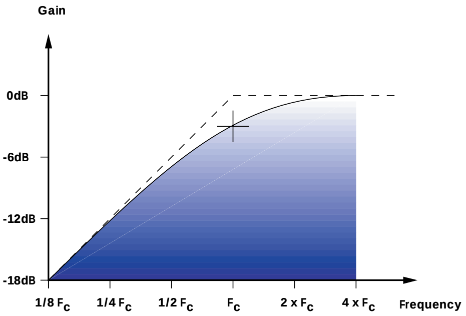


Figure 4: The gain response of a 6dB/octave high-pass filter.

In other words, if you increase the resistance, the cutoff frequency drops. Conversely, if you reduce the resistance, the cutoff frequency rises. And what would a synthesist use to control the potentiometer that lets you do this...? Answer: a knob or a slider. In this case, it's the filter cutoff knob (or slider). Of course, this is an example of a passive filter, and almost every analogue synthesizer filter is active (ie. it includes an amplifier of some sort). But the principle remains the same: by adjusting the value of one or more components in a given filter circuit, you can adjust the cutoff frequency.

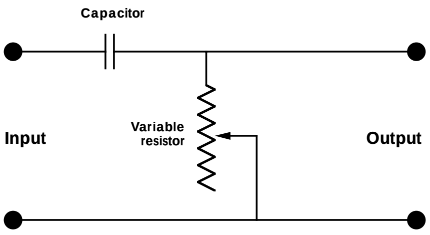


Figure 5: A simple RC high-pass filter.

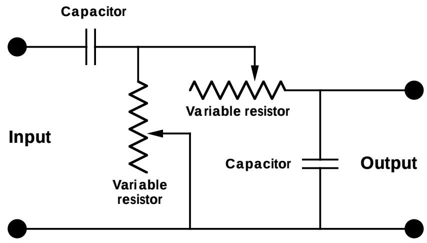


Figure 6: An idealised RC band-pass filter.

The same is also true of a high-pass filter. This is a device that, instead of attenuating high frequencies, attenuates the low ones. You can compare the different frequency responses of low-pass and high-pass filters in Figures 3 and 4.

Figure 5 shows the simplest high-pass filter with a variable cutoff frequency. As you can see, it has the same two components as the variable low-pass filter, but they are swapped around in the circuit. And, once again, the cutoff frequency is simply proportional to 1/R.

## Step 2: More Types Of Filter

Now let's be imaginative. It should be a small leap to see that we can combine a low-pass filter and a high-pass filter to create another type of filter found on many synthesizers: the band-pass filter. This is so called because, instead of attenuating one end of the audio spectrum, it attenuates both ends, permitting just a band of frequencies to pass (relatively) unattenuated. Conceptually it's simple: just place the low-pass and high-pass elements in series, and you get the circuit shown in Figure 6, and a frequency response that looks like Figure 7.

Of course, nothing in life is ever quite that easy, and there are two problems with this approach. Firstly, as we discussed last month, you can't cascade RC filters and achieve the expected result. Secondly, with the same cutoff frequency for both the high-pass and low-pass elements, the signal is attenuated everywhere — by 6dB at its loudest point, and more elsewhere. This would, in most cases, make it unusably quiet.

Figure 7: The band-pass response of a low-pass filter and a high-pass filter in series.

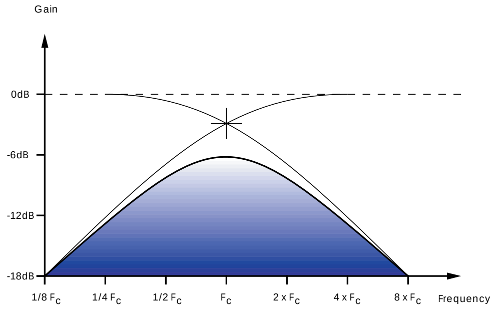
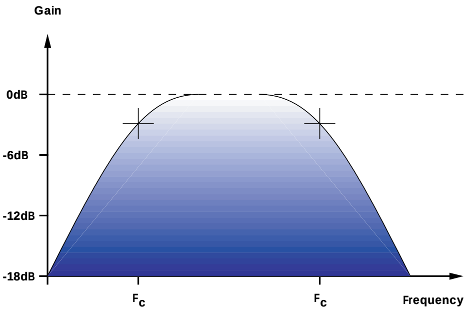


Figure 8: The response of an ideal band-pass filter.



Fortunately, designers can overcome these difficulties very easily. The first solution requires that the input and output impedances are designed to isolate the cascaded stages from each other. The second simply requires that we separate the cutoff frequencies and (preferably) make the slopes steeper by increasing the response of each element to 12dB/oct or even 24dB/oct. The gain response of the resulting filter now looks something like that shown in figure 8.

Extending these ideas further, we can now work out what a band-reject (or 'notch') filter is. Let's take a low-pass filter with a cutoff of, say, 1kHz and pass a signal through it. Now let's pass the same source signal, in parallel this time, through a high-pass filter with a cutoff frequency of, say, 5kHz. When the two are recombined, the frequencies below 1kHz and above 5kHz will come through relatively unscathed, but all those between will be attenuated. Neat, huh? Well, no... the phase shifts introduced by the two separated filters can cause all manner of side-effects when the signals are mixed together again. Nevertheless, the block diagram and the response of our idealised band-reject filter are shown in Figures 9 and 10 (see page 138).

## Step 3: Sweeping The Filter

So, finally, we can make all those interesting filter sweeps and sounds that you've been waiting for? Sorry... but no. Certainly, we've discussed every filter type that you'll encounter on a conventional analogue synth — low-pass, high-pass, band-pass, band-reject, and (in Synth Secrets 4, Aug '99) comb — and we have even seen how we can adjust their attenuation rates and cutoff frequencies. But there are still two vital factors missing from the equation.

The first factor is easily introduced: it's 'voltage control'. If you return to Synth Secrets 3 (July '99) you will find that it is devoted to the idea of changing elements of the sound without human intervention. I won't retread old ground here, but it should be obvious that you can replace the potentiometers in Figures 2, 5 and 6 with a device that responds in some way to the application of an external voltage. In this way, we can sweep the filters' cutoff frequencies by applying modulators such as envelope generators and LFOs — see Figure 11.

But what about the second missing item? Ah, that's a bit more involved...

## Step 4: Resonance

Almost without exception, all physical objects resonate. Or, to put it another way, almost all objects will vibrate naturally at certain frequencies. If the object in question is a stretched string, the lowest such frequency is the fundamental pitch produced when the string is plucked. But what happens if you don't pluck it? In isolation (of course) nothing.

The key word here is "isolation". Let's consider what happens when you place the string in front of a speaker that is reproducing some music. You will then notice that there are times when the string vibrates, and others when it does not. What you are observing is *resonance*. If the string is excited by frequencies in the music that coincide with its natural resonant frequencies, it will vibrate in sympathy with the source. If none of its resonant frequencies are present in the music, it will sit there uninterested. The same is true of, say, the body of a violin, or the air in a pipe. Indeed, the positions and relationships of these resonant frequencies are instrumental (if you see what I mean) in defining the tone of the instrument itself. It's also true of suspension bridges, and civil engineers take great care to ensure that bridges won't resonate in the wind. But what has this to do with analogue filters?

Answer 1: Nothing. Passive RC filters have no resonant frequencies. You can put any signal through one and, no matter how complex the filter circuit, it will just pass or attenuate each frequency according to its response.

Answer 2: Lots. When you combine resistors and capacitors with a third component called an inductor, or use them in active circuits with two or more poles, you can make relatively simple circuits that have a large peak in the response at a particular frequency (see Figure 12).

Furthermore, whereas all passive filters have a gain of less than unity, an active resonant circuit can boost frequencies, making the harmonics at those frequencies louder than they were in the input signal. I'd love to tell you why this is so but, as CEDAR Audio's Research Director, Dave Betts, said to me recently, "It's obvious why it happens. It's the solution to a second-order differential equation." Yeah, well... at least I can tell you which frequencies are boosted. It's those that exist near the cutoff frequency.

Figure 13 (on page 140) shows an idealised response for a resonant low-pass filter. As you can see, the filter still attenuates the frequencies far above the cutoff frequency, but a band around the cutoff is boosted. Furthermore, the low frequencies are slightly attenuated. (For some reason, this is rarely shown in diagrams of this sort.) The width of the resonant peak is described by a parameter called its 'Q'. If Q is low, the peak is broad, but as Q increases, the peak in the filter response becomes more and more pronounced and creates dramatic tonal changes. See Figures 14 and 15 on page 140.

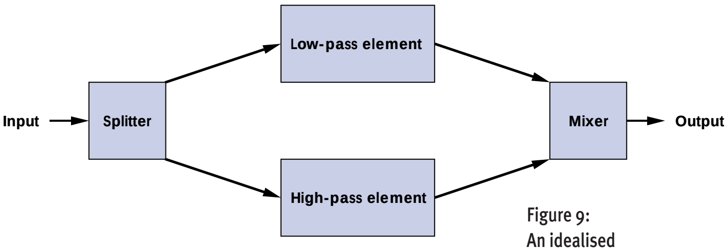


Figure 9: An idealised band-reject filter.

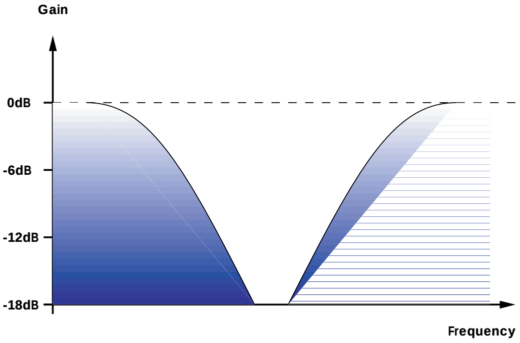


Figure 10: The response of our ideal band-reject filter.

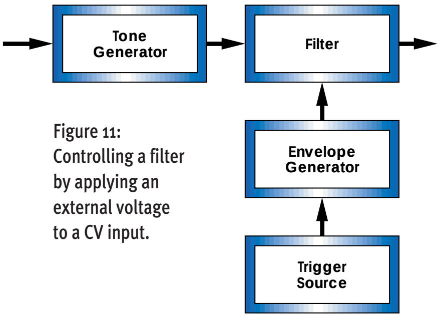


Figure 11: Controlling a filter by applying an external voltage to a CV input.

If you then use a voltage controller to sweep the cutoff frequency up and down the spectrum, the band in which the harmonics are accentuated is also swept up and down. It is this that generates what is, perhaps, the most recognisable and desirable of all analogue synthesizer sounds. Now you can wind up the resonance to create those 'analogue filter sweeps'!

## Step 5: Self-Oscillation

But this is far from the end of the story. If you continue to increase Q, the resonance becomes so pronounced that the high and low frequencies disappear from the signal and another effect occurs: the filter begins to oscillate at its cutoff frequency (see Figure 16). This is a powerful sonic effect, and a filter on the edge of self-oscillation will create a range of unnatural sounds unique to the electronic synthesizer.

## Step 6: Synthesizer Heaven?

More than anything else, the resonant low-pass filter defines the sound of subtractive synthesis. Indeed, some instruments' filters have become so revered that players rarely think about the other facilities offered (or not) by the instrument. This is a narrow view, and one that we must try to expand in

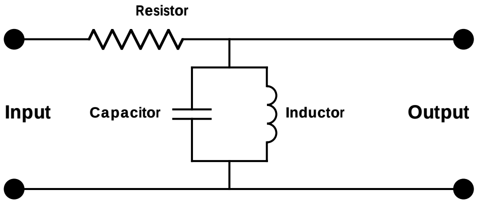


Figure 12: A simple resonant 'LCR' filter circuit.

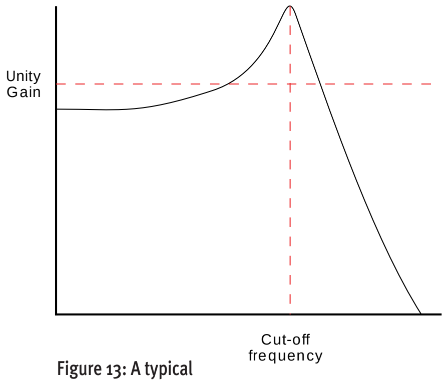


Figure 13: A typical resonant low-pass filter response.

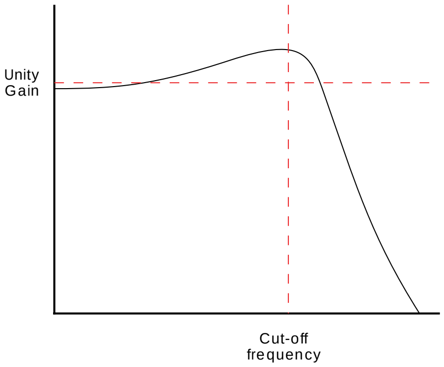


Figure 14: Resonant low-pass filter with low Q.

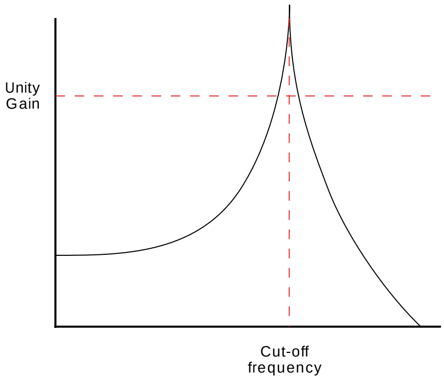


Figure 15: Resonant low-pass filter with high Q.

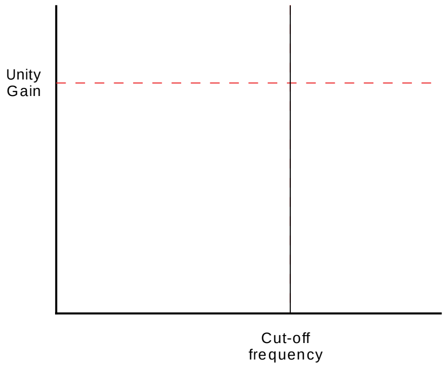


Figure 16: A resonant low-pass filter with maximum Q becomes an oscillator.

later instalments of Synth Secrets. But, in the spirit of all things filter-y, let's briefly touch upon some of the possibilities suggested by Figures 14 to 16:

- You can use a static filter to emphasise certain frequencies, creating character and making a sound stand out in a complex mix.
- You can use two or more static filters to create formants in a sound, and tailor these to imitate the characteristics of the human voice or traditional acoustic instruments.
- If you use a resonant filter with moderate Q and make the cutoff frequency track the pitch, you can create a characteristic 'emphasised' quality that remains tonally consistent as you play up and down the keyboard.
- Increasing Q further, you can (on many but not all synthesizers) enter a region where the filter is on the edge of self-oscillation. It will then ring in sympathy with certain frequencies in the input signal. This creates a distinctive distortion that can be very effective for 'off-the-wall' sounds.
- Increasing Q to its maximum, the filter will become a sinewave generator in its own right. If the filter tracks the keyboard CV accurately, you can then play it as if it were an extra oscillator. You can even add vibrato using a filter modulation CV. In theory, no input signal is passed at this point, but few filters completely remove all the signal, and the result is a tortured sound that has extensive uses in modern music.

If some of these ideas seem alien to you, don't worry. I fully intend to cover them in later articles. For now it's enough to appreciate just how many ways there are in which a filter can modify a signal. And don't forget that, while we have used a low-pass filter response in our resonance diagrams, the same ideas are just as relevant to high-pass filters, comb filters and, to a lesser extent, band-pass and band-reject filters.

So where does that leave us? Look at it like this: a powerful analogue synth offers several filters. The majority of these will be active and, in addition to the ubiquitous low-pass filter, may be selected from any combination of four other types. Most of these will be voltage-controlled, allowing you to manipulate the cutoff frequency using a wide range of modifiers. Many will also be resonant, giving you even more powerful ways to modify and sculpt signals. Some offer a range of cutoff slopes, and most will self-oscillate, offering a further range of sounds, even allowing you to use the filter as an additional oscillator within the synthesizer. The best filters also allow you to control Q using a CV source, giving voltage-controlled resonance. A vanishing small number will even permit you to insert other signals into a feedback loop that helps to generate the resonance effect, and this can lead to even more startling results.

In conclusion, it should come as no surprise that the filters are the defining elements in an analogue synthesizer. Indeed, if you take the output from a Moog oscillator and pass it through a Korg filter, the result sounds like... a Korg synthesizer. Conversely, if you filter a Korg waveform using a Moog filter, the result sounds like a Moog synthesizer. So, while you may get sick and tired of analogue anoraks whinging on about their filters, they do have a point. Filters are crucial, and if you are into creative synthesis, your sound generation will depend upon what you've got and what you do with it. And that's no Synth Secret at all.

# synth secrets

## PART 7: ENVELOPES, GATES & TRIGGERS

You press a key on your synth. It plays a note. That's it, right? Wrong. Gordon Reid explains the role of envelopes, triggers, and gates in this deceptively simple process.

Let's recap. In the first two parts of this series, we discussed the nature of vibrations and the waveforms of sounds. In part three, we looked at the relationship between signals, modifiers and controllers, and became familiar with the concept of an envelope generator, which helps to illustrate this point. Then, over the following three months, we looked at some of the many attributes of filters. You might think that, with all this under our electronic belts, we are now in a position to discuss the secrets of creating amazing synth sounds.

Unfortunately, we have still got a long way to go before we have covered all the background for programming even a simple monophonic synthesizer. For example (and ignoring the relative qualities of the keyboards themselves) we haven't explained why some synths not only sound different, but feel so different to play. So this month we are going to take a detailed look at Gates and Triggers. Along the way, we will discover why the two most revered monosynths of the early '70s — the Minimoog and the ARP Odyssey — can respond so differently when you play them. But before we can do this, we must revisit a topic first raised in part three. We are going to talk some more about envelopes.

If you've been keeping up with this series, you probably think that you understand analogue envelope generators. After all, they're all ADSRs, aren't they? The Attack time determines how percussive a sound is, and the Decay determines how long the sound takes to fall to the Sustain level after the initial accent. After that, the Release controls the time it takes for the note to die away when you release the key. Simple huh? Yeah... too simple. In fact, there are many other envelope shapes, some more complex than ADSRs, some less so. Furthermore, many envelopes are themselves modified by other controls with names such as Amount and Invert. But these will have to remain matters for another instalment of this series. This month, we're going to look at some unexpected ways in which the common envelopes in *your* synth can affect the sounds you make and the way you play.

### Envelopes & Contours

Picture a common waveform, such as a sine or square wave. These have nice, symmetrical shapes that repeat time, after time, after time... Indeed, if they didn't repeat, you wouldn't be able to perceive the pitch of the sound — at best you would hear a click. The common sawtooth and pulse waveforms (which can be, but are not, strictly speaking, symmetrical) also repeat in this fashion. Now, with these images in your head, consider Figure 1, below. This shows a remarkably simple waveform that is neither symmetrical, nor repeated.

Now, remember what happens when we apply the output of this circuit to another part of the synthesizer such as a voltage-controlled amplifier (see Figure 2, below). If the amplifier's gain at any instant is proportional to the voltage shown in Figure 1, the waveform becomes the loudness contour of the sound. If the destination is a voltage-controlled low-pass filter (a VCF) the curve shown in Figure 1 becomes the brightness contour of the sound. Clearly, the contour in Figure 1 can be used as the output from what we usually call an Envelope Generator. OK, so I covered this point in part three of this series, but it bears repeating.

A good definition of an envelope is this: the graph of the way a parameter changes over time is a visual representation of its envelope. But, in a typical synthesizer, you can — at any given moment — change the value of a parameter using any number of modifiers (see Figure 3, on page 130). The total envelope, therefore, may be the sum of the simultaneous actions of numerous devices. So here's this month's first Synth Secret (and it's a biggie!):

*What we usually call an envelope generator may be only one contributor to the true envelope of a given parameter.*

It is for this reason, perhaps, that some of the earliest synthesizer manufacturers adopted a different term for the devices that we now call Envelope Generators. They called them 'Transient Generators'. It's a more precise term, and the one that we will use from now on.

▶

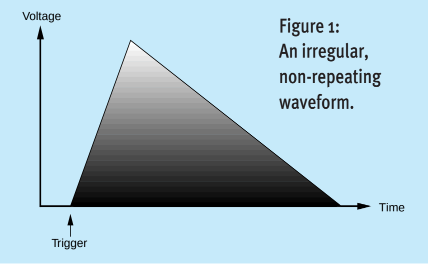


Figure 1: An irregular, non-repeating waveform.

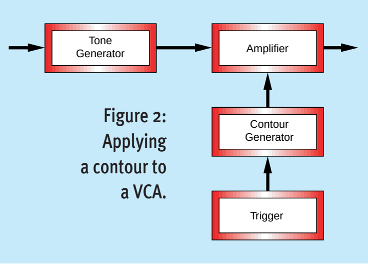


Figure 2: Applying a contour to a VCA.

So now we are in a position to apply some standard terminology to Figure 1. The first stage is the Attack of the transient, and the second stage is its Decay. The figure represents, therefore, a functional 'AD' transient generator. Simple as these devices appear, you should not underestimate the power of AD generators. Figure 4 (opposite) shows what happens when you apply two dissimilar AD contours to a single modifier such as a VCA or VCF. As you can see, the envelope generated by the summation of the two contours is a more complex 4-stage contour. Furthermore, it's one that you cannot obtain from what is commonly called a 4-stage ADSR envelope generator. Food for thought, huh?

## Triggers & Gates

Figure 4 shows what happens when you apply more than one transient to a single parameter at any given time. But that's just one plot in this month's story. There's a second, and it's related to what happens when we play more than one note sequentially...

As you know, most analogue monosynths are controlled by keyboards. What you may not know is that many of these generate three signals every time you press a key. The first is the pitch CV; it helps to determine the pitch of the sound produced, for any given key. The second is the Trigger. This is a pulse of short duration that, at the exact moment you press a key, can trigger the actions of devices such as transient generators. The third is the Gate. Like the Trigger, the Gate's leading edge tells other parts of the synth that you have pressed a key. However (and unlike the Trigger) the Gate remains 'Open' for the whole time that you depress the key. This means that it can also tell the rest of the synth the exact moment that you *release* the key. Figure 5 (opposite) shows these signals. Knowing about Gates and Triggers lets us extend our ideas about transients considerably, and leads us to our next Synth Secret:

*Although a Trigger will initiate a transient generator, it's the Gate that tells the synth to continue developing the contour until the key is released. Without a Gate, all you would hear would be a short 'blip' at the start of the sound, but nothing thereafter.*

Figure 6 (see page 132) shows a 3-stage transient generator that uses the timing information in the Gate signal to complete the Attack stage and then maintain the maximum voltage for the entire duration that the key is pressed. For reasons that are obvious if you studied geometry, we call this device a Trapezoid transient generator. The EMS VCS3 is one of the few synthesizers that offers a Trapezoid, and in the early '70s I thought that it must be a very grand device indeed because it had such a grand name. Oh well...

The next level of complexity is the one that everybody knows; it's the ADSR. In the '70s the 4-stage ADSR contour was so standard that many players called all transient generators 'ADSRs', whether they were or not. Now look at Figures 1, 6 and 7 again. Each of these assumes that every transient exists in isolation, and that every contour has the time to run its course before you initiate the next. But, even on a monosynth, this simply won't be true in the majority of cases.

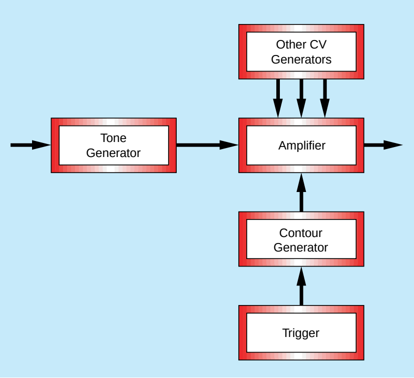


Figure 3: Creating a composite VCA envelope using multiple CV sources.

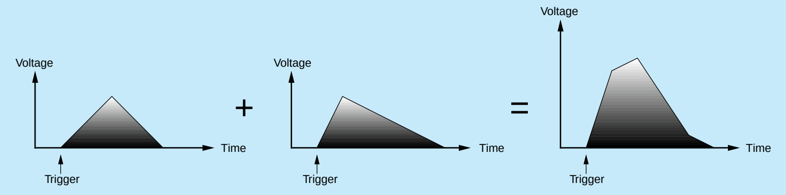


Figure 4: Applying two simple transients to the same modifier.

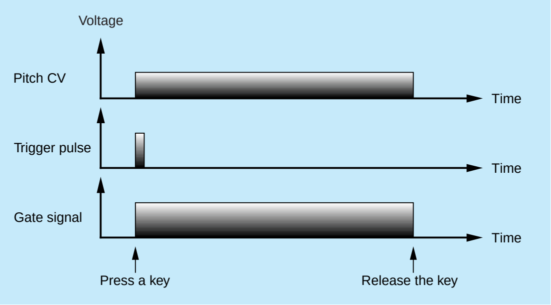


Figure 5: Examples of a Trigger, a Gate, and a pitch CV.

For reasons that should be obvious, we call Triggers and Gates timing signals, and every synth needs them. But why both? Surely the Gate is performing the trigger function, and the Trigger itself is unnecessary? This is a question that dogged me for many years, until I was fortunate enough to have both a Minimoog and an ARP Odyssey sitting next to each other in my 'live' keyboard rig. I often used these to play a heavily contoured solo sound, with an instantaneous (Attack = 0) percussive 'spike' of loudness and brightness at the start of each note. This spike was, of course, created by applying a pair of contours to the VCF and VCA.

Strangely, and despite the outstanding reputation of the Minimoog's contour generators, I knew that I preferred to play my fast solos on the Odyssey, but I did not know why. All I knew was that my playing sounded punchier on the Odyssey, and that I could play at higher speeds than I could on the Minimoog. The reason for this was nothing to do with my playing (I was rubbish on both) nor was it anything to do with the relative quality of the instruments' keyboards. The answer lay in the engineering within the instruments: the Odyssey uses triggers, while the Minimoog does not.

Look at Figure 8 (on page 132). This shows what happens when I play lines very quickly on an Odyssey. As you can see, the notes overlap because I can't remove my fingers quickly enough from previous notes before playing the next. But you can also see that the contour remains dynamic, and that the start of each note is clearly defined. This is because the Odyssey re-triggers its transient ▶

generators every time I play a note, regardless of whether previous ones are released or not, and regardless of whether the Gate is Open or not. This means that my sounds are correctly articulated, no matter how unevenly I release the keys.

Let's contrast this with the Minimoog, a synthesizer that does not generate triggers. Its single, unconventional timing signal is called an S-Trigger, but is, in fact, a Gate. So the same line played identically on the Minimoog produces the filter and amplitude contour shown in Figure 9 (opposite, below). As you can see, the envelope is far less interesting and, although you will still hear the second and successive notes at the sustain level, the punch is lost. The synth solo becomes dull and uninteresting, and the audience remains unimpressed. Indeed, the only way to recapture the dynamics and punch of the Odyssey solo would be to release each note before the next, and I have already admitted that I am not capable of that at high speed. Consequently, for average players rattling off fast solos at the limits of their abilities, the Odyssey is superior to the Minimoog (the fact that the Odyssey keyboard is duophonic and the Minimoog's is monophonic also helps, but that's a topic for another day).

At this point, it's probably worth asking yourself what the result would be if the transient generators on the Odyssey and the Minimoog were merely ADs or trapezoids. The answers may surprise you. If the contours were ADs, the Odyssey solo would sound much the same, but the Minimoog solo would disappear to silence after the first note or two (see Figures 10 and 11, on page 134).

Strangely, if the contours were trapezoids, the two solos would sound identical, provided that the Odyssey's contours did not reset to zero at the start of each note (see Figures 12 and 13, on page 134). Hang on... what's all this about resetting to zero? Check out Figure 14 (also on page 134). This shows the contour of a series of 'reset to zero' trapezoid transients. Some synths reset in this way, and it can lead to a very disjointed sound indeed. Because of their slower Attacks and Releases, this is particularly noticeable on some string ensembles where the single envelope generator resets every time it is initialised. Some early polysynths do this, too. It's horrible, and sounds like the instrument is swallowing its tongue. Glump!

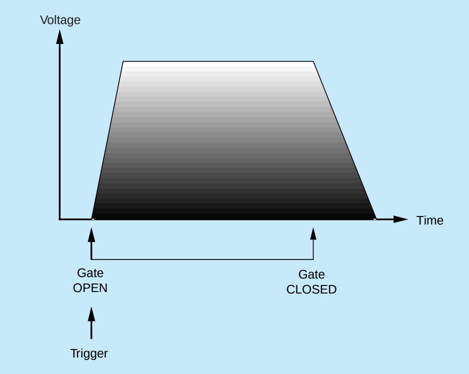


Figure 6: A Trapezoid contour.

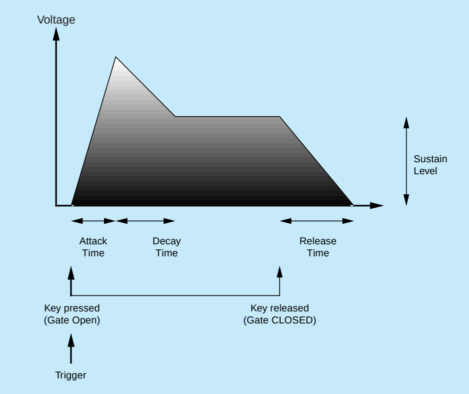


Figure 7: A standard ADSR transient.

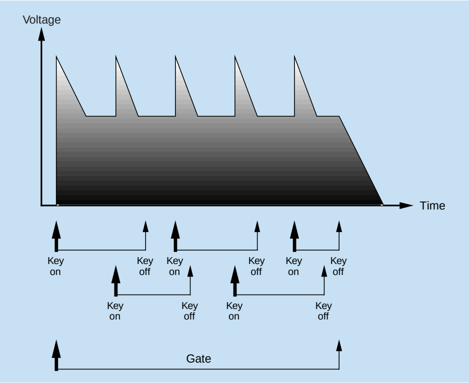


Figure 8: Re-triggering on an ARP Odyssey. The bold arrows show the triggers on each Key On.

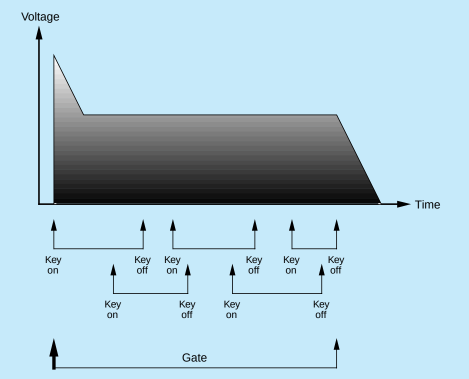


Figure 9: The lack of re-triggering on a Minimoog.

## Putting It Together

We have covered a lot of ground this month, but please don't think that we've done more than scratch the surface of this topic. There are many other facets to transient generators and timing signals. After all, we have only come as far as 4-stage contours, and even analogue synths can have five or more, with Hold stages and Break points amongst other extra goodies. And then there are the synths that allow you to control each stage of the contour using Control Voltages as well as the generator's own knobs or sliders (a common use for this is to make a transient quicker at high pitches than at low ones — just as happens on acoustic instruments such as pianos and guitars). Also, there are other useful destinations for the contours. One of these is the pitch of the sound. Many instruments are slightly sharp at the start of each note, and a simple transient generator such as an AD allows you to recreate this effect. Another useful destination is the modulation speed. 'Real' players do not add vibrato, growl or tremolo with electronic regularity, and changing the effect by applying contours to the LFO speed

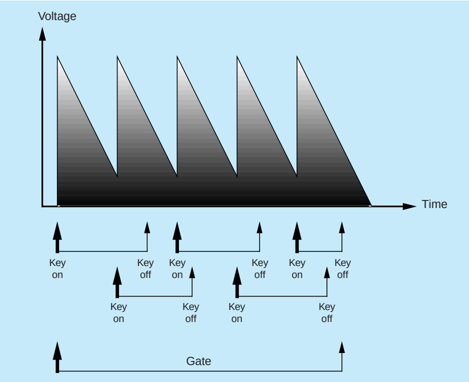


Figure 10: A re-triggered AD transient.

and depth can be most affective (this was one of the rare facilities that made the Yamaha GX1 and CS80 so revered). And what about extending the ideas of applying two contours to a single VCO or VCF, and even delaying the trigger of one slightly with respect to the other...?

Unfortunately, there's no space this month to delve into these issues, so I'm going to finish this article with a little story. When you hear a sound, your brain uses just the first few milliseconds to ascertain what instrument is producing it. Roland was the first manufacturer to take advantage of this, and its (albeit digital) Linear Arithmetic Synthesis included a brilliant trick: a short sample generated the beginning of each sound, with a standard synthesized tone for the Sustain and Release stages. So, if the sample was the 'chiff' of a flute, it hardly mattered (within reason) what came next, the patch sounded pretty much like a flute.

Of course, you can't add samples to the sounds on an analogue synthesizer. But if your synth can recreate the initial waveform of a flute (it's only a filtered sawtooth plus a little noise, after all), it follows that you need only use a transient generator (or two, or three...) to recreate the initial contours of the flute. Then you will be able to play a single note that sounds very much like a flute. And that's exactly how it is.

So this month's final Synth Secret, and the one that explains why some Contour Generators are calibrated in milliseconds rather than just '0' to '10' is:

*If you want to recreate acoustic sounds or program interesting synthesizer sounds deterministically (rather than rely on blind chance) then, at the very least, you must be able to analyse and create the brightness and loudness contours of the sound.*

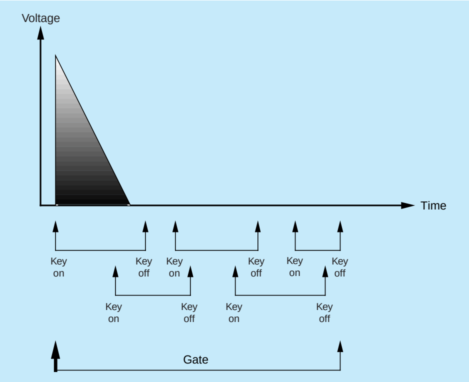


Figure 11: The failure of the AD transient with no triggers.

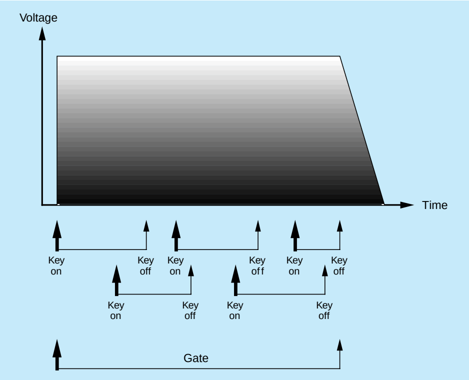


Figure 12: A re-triggered trapezoid transient.

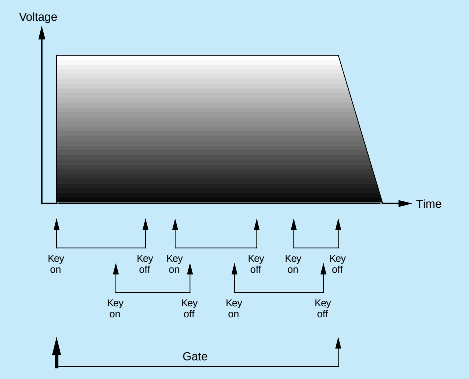


Figure 13: A set of gated trapezoid transients.

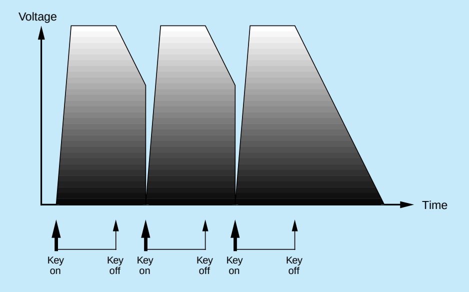


Figure 14: A contour generated by a 'reset to zero' transient generator.

But will you be able to use your wonderful flute patch to play a complete line or phrase in a way that sounds like a flute? If you are using an Odyssey (the decidedly superior synth a few paragraphs ago) the answer is "probably not". After all, no flautist chiffs every note individually, so permanent multi-triggering is a disadvantage. The Minimoog, on the other hand, allows you to retrigger some notes (by releasing the previous ones) and slur others (by overlapping them) much as the real flautist would do. When some people realise this — that particular synths are more suitable for certain tasks than others — they start collecting synths. In this case, the Minimoog is capable of producing a more realistic flute solo — but of course, you could always hunt down one of the rare instruments with a single/multi triggering switch, and then you would have the best of both worlds... If you start to think like this, watch out. A room full of synths is but an understanding bank manager away! SOS

# synth secrets

## PART 8: MORE ABOUT ENVELOPES

Gordon Reid reveals some of the limitations of the 'classic' ADSR envelope with reference to a practical synthesis example, and explains some of the different types of envelopes found on 'classic' analogue synths, from AR envelopes right up to highly flexible digitally controlled EGs.

I mentioned at the end of the last part of this series that people often start collecting synths when they begin to appreciate that some are better suited to certain tasks than others. As your understanding of synths grows, so too can your collection. So, imagine now that you're standing in the midst of just such a carefully assembled collection of analogue classics. What would you find there? Undoubtedly plenty of Sequential gear; a Prophet 5, say a couple of Prophet 10s, a Prophet 600 and a Prophet T8. And Rolands, of course; there'd probably be several A-frame stands sagging under the weight of Jupiter 8s and 6s, and JXs of all descriptions. You'd have to have something from Oberheim's heyday in there too, like an OB8, and maybe a Memorymoog to represent the work of Dr Robert's pioneering company. Then there'd be a few oddities, like rare analogue Kawais, and a Crumar Spirit. No need to stop at keyboards, either. You'd have to include various Roland MKS-series rack synths, like the MKS30, -70, and -80, plus maybe Crumar Bits and Cheetah MS6s.

What a selection! Surely, with classics like these, there's not a sound in heaven or on earth that is beyond your synthesis capabilities? Well, actually, the ugly truth is that there are *plenty* of very common sounds these so-called synthesizers can't, um, synthesize — and one of the major reasons for this has to do with their contour generators (or envelope or transient generators, as you may prefer to call them). You see, every one of the analogue synths in the list above (and most others besides) uses the ADSR contour generators we discussed in parts 3 and 7 of this series. And, flexible as these are, there are many, many sounds that you can't imitate using 4-stage envelopes. But don't just take my word for it; let's consider what's necessary to synthesize a fairly common-or-garden sound and you'll soon see what I mean.

### Bold & Brassy

Imagine you're trying to synthesize a realistic spit brass sound that goes 'psst' at the start and then swells to its full glory. Just by using our ears, we know that sounds such as these start out from silence, and exhibit a pronounced 'spit' at the start of the note. They then drop back to a significantly lower level and more muted timbre before swelling slowly to their full volume and brightness. Finally, notes decay quickly to silence once the players stop blowing. The volume and brightness contours of any such sounds will therefore look something like the curves shown in Figure 1 (see below).

Now, let's look at the common 4-stage ADSR envelopes offered by all the synths in the 'classic' collection mentioned above (see Figure 2, below) and ask ourselves why we can't use these to recreate the desired brass sound. The problem lies not only in the number of stages offered by an ADSR contour generator, but in the large number of other limitations it imposes upon us. Let's try to work out how many of these there are:

Firstly, if you have an ADSR envelope, the voltage at the start of the contour has to be zero. This isn't a problem if you want to synthesize a spit brass sound, but it could prove a limitation with other types of sound. Secondly, with an ADSR, the Attack phase of the contour always moves in a positive direction. Thirdly, the ADSR always reaches its maximum level at the end of the Attack. Fourthly, the Decay is always negative with respect to the Attack. Fifthly (is there such a word as "fifthly"?) the Sustain Level always begins at that level reached at the end of the Decay. Sixthly (now we are definitely on shaky grammatical territory)

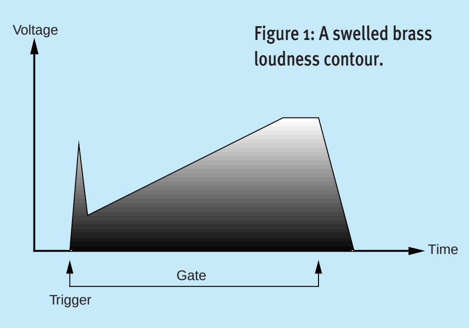


**Figure 1:** A swelled brass loudness contour.

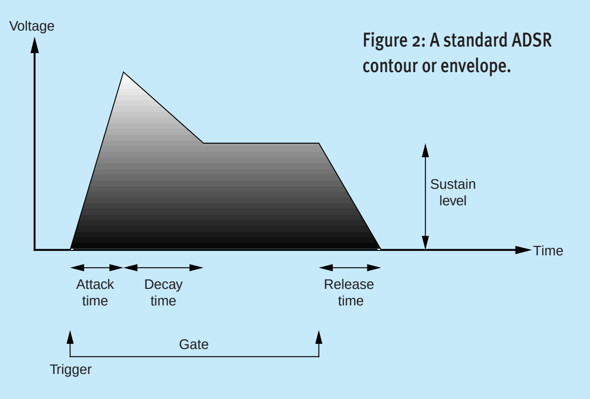


**Figure 2:** A standard ADSR contour or envelope.

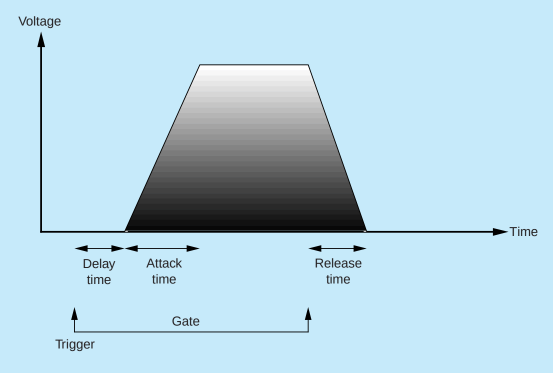


Figure 3: The Korg MS20 'Envelope Generator 1' contour generator.

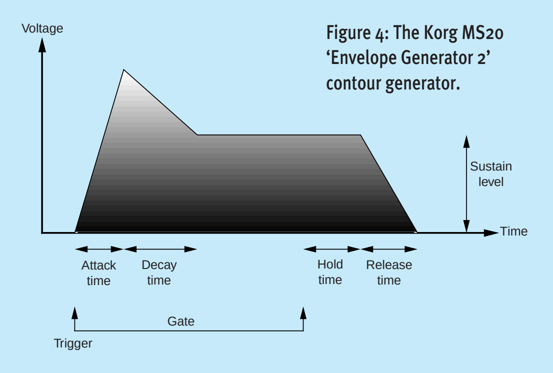


Figure 4: The Korg MS20 'Envelope Generator 2' contour generator.

the Sustain Level is a constant voltage. Seventhly, the Release starts at the Sustain Level. Eighthly, the Release always moves in a negative direction. Ninthly, the Release always ends up back at zero.

Of these, it's the third and fifth limitations that are most damaging to the spit brass sound shown in Figure 1. This is because the level at the end of the Attack stage is *not* the maximum, and the Sustain Level is *not* the level at the end of the Decay! So, is that the end of the story? If the most revered synths in analogue history use ADSR envelopes, must we forever live without synthesized brass sections?

## Back To The Classics

Of course, this is not the case — and if you read last month's instalment of this series, you'll know of one workaround already (described in principle last month at the top of page 130). Using a Control Voltage mixer on a modular synth, you can combine the voltage contours of several simple envelope generators to create more complex multi-stage envelopes, and these could be used to recreate the spit brass contour, for example. However, on an analogue modular, you'd be hard-pressed to create a usefully polyphonic sound.

Fortunately, there are a number of non-modular synths that do not just offer straightforward ADSR generators. One of the most common of these is the Korg MS20. This has two contour generators, one with five stages, the other with three. The simpler of these is a DAR envelope: Delay, Attack and Release. Figure 3 (above) shows this.

As you can see, this lacks the Decay stage and

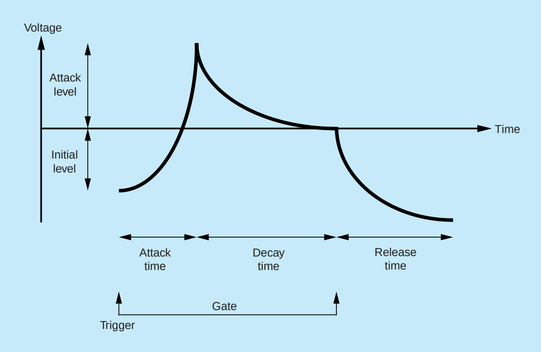


Figure 5: The GX1 filter contour generator.

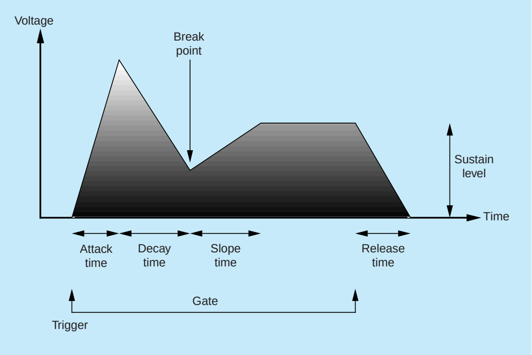


Figure 6: A 5-stage ADBSSR contour from the Korg EX800.

user-programmable Sustain Level (the Sustain is always the maximum voltage). But, by way of recompense, the DAR allows you to program a Delay that determines a length of time before the contour is initiated after the start of a note. This is particularly useful, for example, when creating delayed vibrato (we'll come back to vibrato in a couple of months).

Anyway, the second MS20 contour generator is even stranger than the DAR. This is an ADSHR, where the 'H' stands for Hold. If you look at Figure 4 on page 63 you'll see that the Sustain Level is... um, sustained for a time after the Gate is released. A common use for this would be to hold a note well after you release the key, maybe while you prepare your hands for the next task at hand (remember, Korg released the MS20 before sustain pedals were common on monosynths, and well before MIDI or many of the other performance facilities that we now take for granted). Despite this, the MS20 is clearly incapable of generating the contour from Figure 1.

Maybe we should look back a little further in Korg's history. The Korg 700, 700S, 770 and 800DV were perhaps the strangest monosynths ever produced. These shared a style of contour generator that is hardly recognisable in today's homogeneous world. With just two sliders sharing the names Attack/Slow, and Percussion/Singing, and with switches for Sustain On/Off, or (depending upon model) Percussion/Sustain/Hold, these were nonetheless capable of all the common ADSR contours, although with less control over the Decay and Release times. Unfortunately, despite their quirks, these were also unable to generate the type of contour we're seeking.

Hmm... perhaps we should look elsewhere for our spit brass patch. Surely the two greatest Yamahas of all time can do what we want? How about the mighty GX1? For no less than £40,000 in 1975 (well over £250,000 at 1999 values) you might have expected something a little special. Well... yes and no. Each of the GX1's voices had two contour generators, one permanently patched to the amplifier, the other to the dual high-pass and low-pass filters. But, yet again, the loudness contour was just a conventional ADSR! The filter contour was, however, different from anything we have discussed so far. This offered an Initial Level control, an Attack Level control, plus more conventional Attack, Decay, and Release times (see Figure 5 above). It looks superficially similar to the ADSR, but has a number of significant differences. For example, the Initial Level is not necessarily the minimum level at which the Release stage terminates. Also, there is no defined Sustain stage — the Decay decreases exponentially to zero volts, which (at least, on long notes) is the level until the key is released. Nevertheless, the GX1 is no more capable of the 'spit brass' sound than any of the other synths discussed so far. Not surprisingly, the mighty CS80 (the cheaper 'son of' the GX) shares this structure. Powerful it is, but not all-encompassing. So where do we turn now?

How about the best-known analogue of all, the Minimoog? No... that's just got a pair of ADSD contour generators, wherein the Release time equals the Decay time, or is zero. So what about the ARP Odyssey and its soul-mate, the Octave Cat? No, again. These have an ADSR and a simple AR. In fact, the solution to the problem lies with none of the so-called 'classic' analogues...

## From Dinosaurs To Digits

All the instruments we have mentioned so far this month used simple knobs or sliders to change resistances within, and therefore control the responses of, their circuits. These responses could be, for example, the pitches of the oscillators, the cutoff frequencies of filters, or the time constants in various stages of their contour generators.

Nevertheless, lots of these synths were analogue/digital hybrids that used microprocessors to control many of their memory and synthesis functions. Sure, they had knobs, but they also had analogue-to-digital converters that translated the voltages they controlled into numbers. These numbers were converted back into voltages to generate a sound, but it was the numeric form that the instrument saved in its memory, and which you could recall at the touch of a button.

As you know, microprocessors handle numbers as a series of 0s and 1s known as a binary number. The number of digits used in this number is the number of 'bits', and it is this that determines the accuracy with which you can resolve any given parameter. Many hybrid synths used five bits to describe important values, thus offering just 32 possible values. Even the better and more expensive digitally controlled analogue synths used only seven bits, offering just 128 possible values for the parameter. This was quite a limitation — whereas the original knob may have been capable of thousands of discrete settings, the microprocessors constrained you to just a handful of values. Indeed, the memory chips used in early synths were very expensive, so manufacturers were keen to use as few bits as possible. By doing so, they could offer more patch memories for the same number of chips, and minimise the battery power needed to keep the memory 'alive' when the keyboard was switched off.

Of course, it didn't take long for manufacturers to realise that players didn't need dozens of knobs to program these machines. Or, to be more cynical, it didn't take long for manufacturers to realise that they could cut their costs if they removed all the expensive knobs and sliders. Instead, they could assign an identification number to each sound parameter, and it then became the player's job to request the appropriate parameter before editing it using a single knob or a pair of Up/Down buttons. Consequently, the mid-'80s became the heyday of Digital Parameter Access (DPA) synths, from the cheap-and-cheerful Korg Poly 800, to the... well, cheap-and-cheerful Roland Alpha Junos. And believe it or not, it is these synths that provide the solution we've been looking for.

## And The Answer Is...

It's a strange quirk of fate that the transition from purely analogue to digitally-controlled-analogue architectures made possible many of the facilities that we take for granted on modern instruments. Freed from the constraints of expensive control panels, manufacturers were able to extend the specifications of many parts of the synthesizers' architectures.

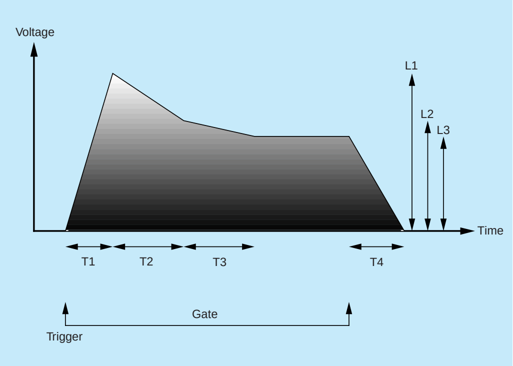


Figure 7: The Roland Alpha Juno 2 5-stage contour generator.

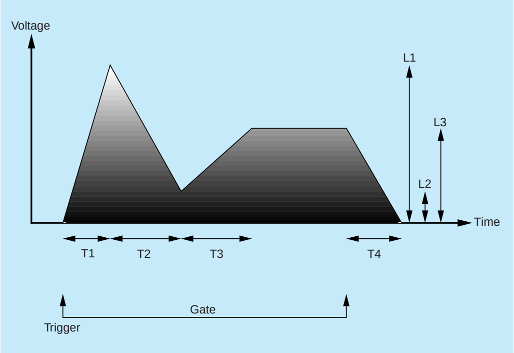


Figure 8: Another Alpha Juno contour.

For example, the Korg Poly 800 and its rackmount sibling the EX800 offer no fewer than three contour generators, each with six parameters controlling five stages (see Figure 6 on page 64).

On the Poly 800 and EX800 these are called DEGs — Digital Envelope Generators — because their shapes are calculated in real time by a microprocessor before being converted into analogue voltages. As you can see from Figure 6, these DEGs offer a far greater range of contours than the simple ADSR, if only because the Break Point can be above or below the Sustain Level. Nevertheless, most of the limitations described earlier in this article still apply. In particular, the maximum level still occurs at the end of the Attack stage. Furthermore, the Poly 800 and EX800 are two of the DPA synths that use just 5-bit words to store their parameter values, so each of the six parameters in the contour can take a value only between 0 to 31. Consequently, if you want an Attack time that seems to lie, say, somewhere between 18 and 19, you're stuffed.

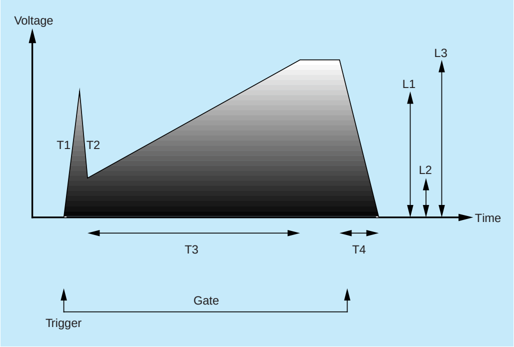


Figure 9: The Alpha Juno 'spit brass' contour.

So, finally, let's turn to the Roland Alpha Juno series, a family of synths that receive more than their fair share of opprobrium, and which are grossly underrated by almost every analogue synth fanatic on the planet. A number of things make these little Rolands special, and one of these is their contour generator. This allows you not only to determine four time settings, but no fewer than three levels, making it dramatically superior to the three time settings and one level of the ADSR. As you can surmise from Figure 7 (see above), the Alpha Juno has no difficulty creating the basic ADSR shape — and before we move on, there is an important lesson here. By choosing the parameters of a 5-stage contour carefully, you can make it look similar or even identical to a 4-stage ADSR. In fact, however many stages the contour generators on your synth have, you can usually recreate the contours of a envelope generator with fewer stages. Sure enough, if you set the Sustain Level of a 4-stage ADSR to its maximum, you always generate a 3-stage trapezoid contour, no matter what the A, D and R values may be. Similarly, setting the Sustain Level to zero and making the Release Time equal to the Decay Time recreates the 2-stage AD and AR contours.

But, of course, the extra stages and levels are there to allow you to create new contours that take us well beyond the capabilities of ADSRs — and you can see two of these in Figures 8 and 9 (left). As you can see, Figure 8 duplicates the 5-stage contour of the EX800. This is a significant advance over the ADSR because we have used L2 (Level 2) as the Break Point in this contour. But we have still overlooked the 'L1' parameter, the one that allows the level at the end of the Attack to be other than the maximum amplitude of the contour. So let's reduce L1 and adjust the other parameters to create Figure 9. Lo and behold... it's our spit brass contour from Figure 1!

So, here's this month's first Synth Secret: *Increasing the complexity of the contour generators adds many possibilities for more detailed sound creation, without precluding the creation of simpler sounds.*

## Why Are We All Here, Anyway?

Finally, I would like to point out that the backlash against Digital Parameter Access has been the fuel behind the current 'retro' craze for all things knobby and controllable. Perversely, many of the over-priced fashion statements that cash in on this craze ("my Jupiter 6 and Prophet 600 are analogue man, real music, not like that digital rubbish...") have digital memories and quantised parameters, and are therefore analogue/digital hybrids. This means that their parameters are limited by the resolution of their processing systems. Furthermore, many of the most sought-after analogue synths use their micro processors to generate their LFOs and envelopes digitally and, in many ways, are barely analogue at all! If you're interested in synthesis, the best way to deal with the digital vs. retro debate is to ignore it, and keep in mind that the reason these instruments exist is to craft different sounds and make music with them. In short:
*Don't become carried away by the current craze for vintage synths or their DSP-generated descendants. Think about the type of sounds you want to generate, and choose your instrument carefully so that you can produce them.* SOS

# synth secrets

## PART 9: AN INTRODUCTION TO VCAS

Having laid bare the inner workings of oscillators, contour generators and filters, Gordon Reid turns his attention to something which at first sight seems entirely self-evident. Can the humble voltage-controlled amplifier really hold any Synth Secrets?

In the course of this series, there's one synthesizer component I've mentioned frequently — especially in Synth Secrets Part 3, July '99 — without fully explaining what it is. It's the Voltage Controlled Amplifier, or 'VCA', and it is an important element in the operation of any analogue synthesizer. Indeed, so fundamental is the VCA that it has its counterparts in every type of synthesis: digitally controlled analogue synths have 'DCAs', while pure digital synths and samplers have all manner of time-variant amplifiers ('TVAs'), operator levels, and heaven knows what else. So this month's article will focus on VCAs and the ways in which they and their latter-day equivalents help you to create the sounds you want to hear.

### Two Types Of VCA?

Firstly, we must differentiate between amplifiers in the audio signal chain and those used to modify control voltages. So let's start by returning to Synth Secrets Part 3 for a moment, and revisit Figure 1 of that article (Figure 1 of this one, too, opposite). This shows a simple sound generator comprising a tone generator and an amplifier that lets you hear what the generator is... umm, generating. Clearly, the amplifier in this diagram is acting in a similar way to the amps in your hi-fi or car stereo. To put it bluntly, it's making the signal louder.

This is all very well, but it's not a good enough description for us today. Have you ever asked yourself, 'What is an amplifier?' If you are a guitarist, you might describe an amplifier as something you use to add distortion and other effects as you boost your sound to a screaming climax. If you are a hi-fi buff, it's more likely that you will describe it as a huge, control-free, and expensive lump of hand-picked valves and power supplies that make the signal louder with as few audible side effects as possible. But if you were an engineer, you would probably describe an ideal amplifier as a device that changes an input signal of amplitude $A_1$ into a new signal that has the same shape as the original, but with output amplitude $A_2$.

Clearly, if $A_2$ is greater than $A_1$, the signal is louder than before. If $A_2$ is less than $A_1$, the signal is quieter than the original. Simple, yes? But it's still not enough to say that signals become quieter or louder. We need to know by how much they do so. Defining this is also simple: we just calculate the ratio of $A_2$ to $A_1$ and call the result the 'Gain' of the amplifier (Equation 1, opposite). So, for example, if $A_2$ is double $A_1$, the ratio is '2' and we say that the Gain is equal to 2. Unfortunately, we can't assume that you will perceive the signal to be twice as loud as before, because the human ear does not work this way... but that's a discussion for another time.

Now let's think about the volume control on the front of a device such as a cheap transistor radio. In human terms this is increasing and decreasing the volume of the signal that you hear, so it is modifying the Gain in some fashion. Figure 2 below shows a possible implementation for such a control.

In this diagram, the receiver circuit produces a low-amplitude signal which passes directly to a preamplifier. This boosts the signal to 'line level'. The output from this then passes through a volume control which, in this implementation, is just a passive potentiometer. If the volume knob is turned fully clockwise, the signal is unaffected, and passes at line level to the power amplifier. This then boosts the signal to a level at which it can drive a coil of wire and lump of cardboard (a speaker) so that you hear the broadcast. If, however, you turn the volume control progressively anticlockwise, the loudness of the sound you hear will steadily diminish until, when the knob is fully anticlockwise, silence will reign. This is because the potentiometer is progressively reducing the amount of signal reaching the power amplifier.

It might seem odd, but you could redefine this ▶

$$G = \frac{A_2}{A_1}$$

Equation 1: The Gain 'G' of an amplifier.

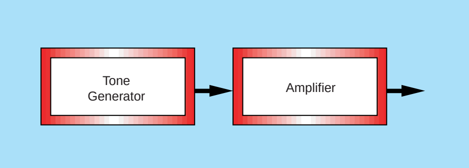


Figure 1: A simple sound generator.

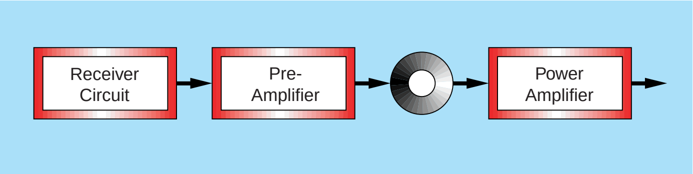


Figure 2: A simple radio receiver.

$$G = \frac{\text{Output from Power Amplifier}}{\text{Signal produced by Receiver}} = G_{PRE} \times G_{ATTEN} \times G_{POWER}$$

Equation 2: The Gain of the whole system.

volume knob as an amplifier of sorts. It's just that its Gain never exceeds unity. When it is 'off', the ratio of Output/Input equals 0, and when it is set to its maximum the ratio of Output/Input equals 1. By definition, therefore, the Gain always lies between 0 and 1. Normally, however, you would not call a passive device of this sort an 'amplifier'. You would call it an 'attenuator'.

There's another important point to consider about the amplifier/attenuator chain in Figure 2. We now know that, for any position of the knob, the attenuator's Gain is some number $G_{ATTEN}$ which lies between 0 and 1. We can also assume that the preamplifier has a large Gain (we'll call this $G_{PRE}$) and that the power amplifier has another large Gain (which we'll call $G_{POWER}$). We can then say that the total Gain through the system is the product of all the individual gains, as shown in Equation 2, above.

This is an important result, so let's call it a Synth Secret:

*Whenever you have more than one amplifier and/or attenuator in series, you can calculate the Gain of the whole system simply by multiplying the individual Gains together.*

## A Better Circuit

Although Figure 2 is easy to understand, the circuit it depicts is not a good one. This is because the audio signal passes through the attenuator itself. Since this will usually be a cheap potentiometer, it's likely that it will introduce crackles and distortion into the signal. This, for most people, is something to be avoided, so we need a better circuit that performs the same job without the unwanted side effects.

Figure 3, above, shows such a circuit. This contains the same elements, but the volume control knob now attenuates a voltage source that controls the Gain of the preamplifier. In other words, we have redefined the preamp as a Voltage Controlled Amplifier, and the audio signal no longer passes through the volume control.

## From Radios To Synthesizers...

Let's now relate our 'radio' to the structure of a simple analogue synthesizer. Clearly, the signal generated by the receiver circuit could be anything: speech, Beethoven's 5th, Silverchair's 'Freak', or a sawtooth wave. So why don't we replace the words 'receiver circuit' with the words 'tone generator'? Next, let's consider the power amplifier in the diagram. A few synths with built-in speakers have these (for example, the ARP 2600, the Roland HS60 and the Yamaha YS200) but most leave the final amplification to external units. As a result, we can lose the 'power amplifier' from our block diagram without compromising our discussion. That leaves the preamplifier and volume control.

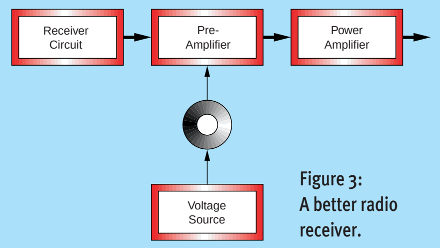


Figure 3: A better radio receiver.

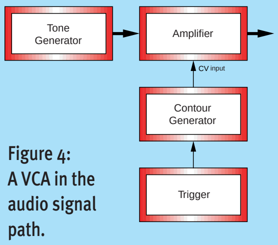


Figure 4: A VCA in the audio signal path.

If you think back to Synth Secrets Part 3, you'll remember that we can replace a volume control with some sort of controller circuit. So Figure 4 shows the same audio signal path as before, but the preamplifier (now simply called 'Amplifier') is controlled by a contour generator which is itself being triggered by... well, a trigger. Figure 4 may look very different from Figure 3, but in essence it is describing the same relationship of generator, amplifier, and volume control. So let's analyse what's happening.

Let's say that the Tone Generator produces an initial signal of ±2V. Let's also say that the contour generator has been designed to output an ADSR envelope which ranges from 0V to +5V. Now let's suppose that the VCA outputs nothing when 0V is presented to its CV input, and it outputs a maximum audio signal amplitude of ±10V when a +5V CV signal is presented to its CV input. This means that the amplifier in has a maximum Gain of 5 (G=10V/2V) and a minimum Gain of zero (G=0V/2V). Finally, let's define that the amplifier's response is 'linear', ie. that 1V at the CV input generates a Gain of 2; 2V at the CV input generates a Gain of 4, and so on. This means that the amount of audio signal Gain at any given moment is proportional to the instantaneous level of the contour applied to the amplifier's CV input.

All these voltages may seem confusing written down in this way, so I have represented them (I hope) more clearly in Figure 5.

"Whenever you have more than one amplifier and/or attenuator in series, you can calculate the Gain of the whole system simply by multiplying the individual Gains together."

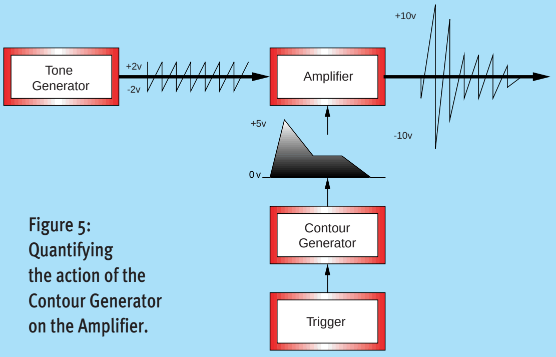


Figure 5: Quantifying the action of the Contour Generator on the Amplifier.

## Initial Gain

You might think that there is enough detail here to represent everything that you need to know about a practical VCA. But there isn't. If you look at the front panels of an ARP Odyssey or an ARP 2600, you'll see a slider marked 'Initial Gain' or 'VCA Gain'. This adds an initial CV, or 'offset', to whatever CV the Contour Generator produces (see Figure 6, right).

So, for example, if we take the contour shown in Figure 5 and add an Initial Gain of +3V, we obtain the contour shown in Figure 7 (top right). This offset has an immediate effect — the VCA is always being told to produce a Gain greater than zero. Because the CV presented to the VCA has a permanent +3V offset, therefore, signal is always produced at the output: provided the filter is open, the synth will produce sound continously until it is switched off.

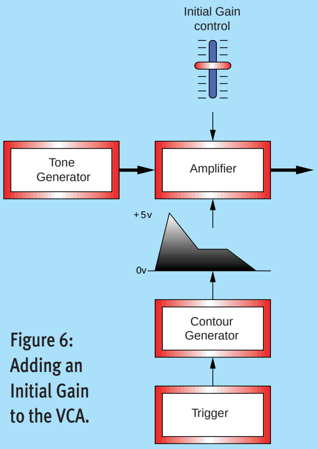


Figure 6: Adding an Initial Gain to the VCA.

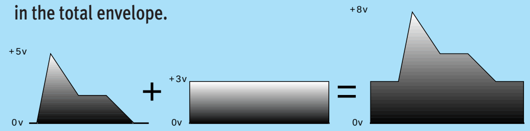


Figure 7: Initial Gain provides an offset in the total envelope.

## Let's Not Go Too Far...

Throughout this discussion we have assumed that the VCA has infinite headroom. This means that, no matter how much amplification you demand from it, it will continue to deliver the same shape of signal without adding distortion or any other artefacts. This, of course, is impossible, and if you ask a VCA to generate a signal beyond its capabilities, a particular type of distortion will result.

Let's look again at Figure 5. You'll remember that we defined the maximum output from the VCA as ±10V, achieved when the controlling CV reached +5V. So, what would happen if we replaced our 0V to +5V ADSR contour with the +8V contour shown in Figure 7? Clearly, at the peak of the contour, the VCA would try, and fail, to generate a signal of ±16V. Since it can't exceed ±10V, the signal is 'clipped', as shown in Figure 8, opposite.

If you look closely at the part of the output waveform that occurs during the period of the ADSR contour, you can see that the output no longer has the sawtooth shape of the original input signal. The 'tops' of the waveform are squared off in the Attack and Decay stages by the amplifier's inability to amplify above its ±10V limit. The result of this is a harsh distortion ('clipping distortion') that disappears as the signal demanded of the VCA returns to a range within its capabilities, and the waveform settles back to its original sawtooth shape. Of course, some players use this distortion creatively, and it can be rather effective when an amplifier clips in a 'softer' way, rounding off the waveform rather than squaring it so dramatically. This line of thought could also lead to a discussion of analogue tape compression and saturation, and one of the reasons why analogue recordings differ from digital ones. But, once again, we'll have to leave this for another day.

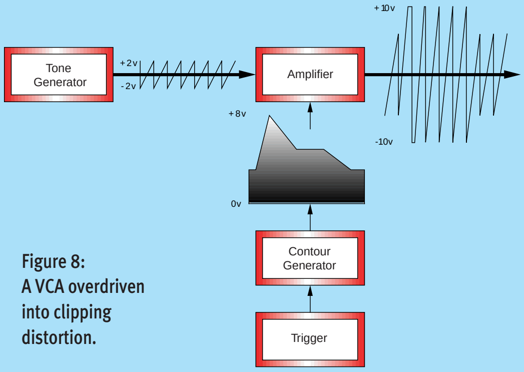


Figure 8: A VCA overdriven into clipping distortion.

## A Different Use For VCAs

Everything we have discussed so far assumes that the VCA is in the audio signal path. But in reality, the majority of VCAs do not reside here: they're in the control voltage paths within the synthesizer.

Let's return to the contour generator controlling the amplifier in Figure 6. You'll remember that this outputs an ADSR contour with a maximum of +5V at the end of the Attack stage. Now, if you think back to last month's Synth Secrets, you will also recall that the vast majority of analogue contour generators give you no control over the level at the end of the Attack: no matter what the settings for A, D, S or R, the Attack Level is always +5V (or whatever that particular device's maximum might be). This, as we have already discussed, will generate a gain of 5 in our VCA. But there are many occasions when we will not want the contour to affect the signal so dramatically. So what can we do if we don't want the result to be so extreme?

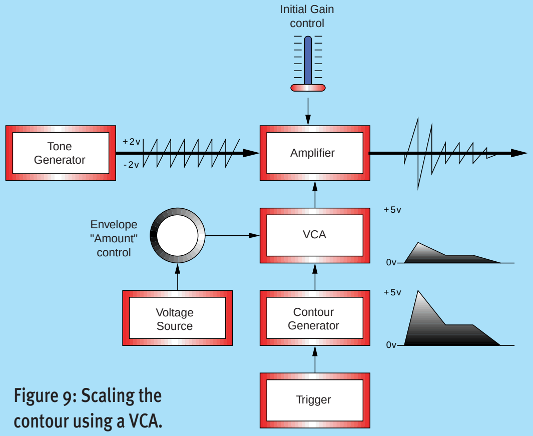


Figure 9: Scaling the contour using a VCA.

Look at Figure 9 above. As you can see, I have placed a VCA in the control signal path, and this is in turn controlled by a CV that is itself controlled by an attenuator. This VCA applies a Gain (determined by the position of the attenuator) to the ADSR contour so that you can attenuate or amplify it without changing its shape.

This result would not be important if the ADSR was the only CV affecting the signal amplifier. After all, reducing the amplitude of the contour would be no different from reducing the final output of the synth, maybe just by turning down the external amplifier. But the ADSR is not the only CV. As well

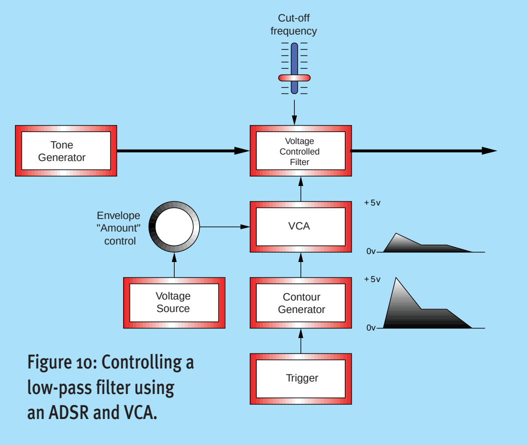


Figure 10: Controlling a low-pass filter using an ADSR and VCA.

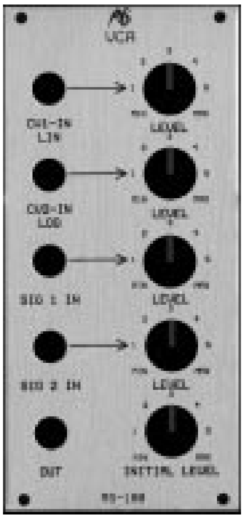


Fig 11: A real VCA module.

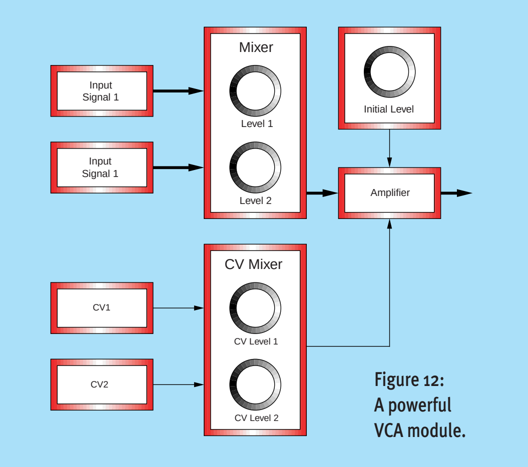


Figure 12: A powerful VCA module.

as the Initial Gain shown in Figure 9, you could have LFOs or a host of other controllers modifying the audio signal amplifier's action upon the signal. In this case, you are changing the degree by which the ADSR modifies the signal *relative* to the initial level and any other modifiers in use.

To make this clearer, let's use another example that you will have encountered more often. Figure 10 is similar to Figure 9, except that the CVs are controlling a low-pass filter, not an amplifier. As you can see, the fader at the top of the figure is now the initial level of the filter, most usually called the 'cutoff frequency', and the VCA is controlling the amount of contour applied to this. Clearly, you don't want the filter to open completely for every sound you make, so almost every synth allows you to attenuate the contour using a VCA, as shown.

## Real VCAs

The important role played by VCAs is often overlooked when we think about synthesis. Indeed, if you look at the control panels of many analogue synths, you'll see that the section described as 'Amplifier' or 'VCA' is most often dominated by an ADSR contour generator, with perhaps an envelope level control (also called 'Amount') and/or Initial Level control too. This is why many novices find it difficult to differentiate between the contour generator, the VCA in the CV path, and the audio signal amplifier itself. Similarly, the filter section will often contain a second contour generator and another 'Amount' control alongside the cutoff and resonance knobs. This means, of course, that there are VCAs in the VCF section too.

Let's finish this month by looking at a well-specified VCA from a British modular synth manufacturer. As you can see from Figure 11 (above right), the device has four inputs, five knobs, and an output. There are two signal inputs (marked SIG 1 IN and SIG 2 IN), which means that there is a signal mixer in the module. There are two CV inputs which control the amplification of the VCA itself, meaning that there is a CV mixer in there, too. The CV inputs are marked CV1-IN LIN and CV2-IN LOG. Each of the four inputs has an associated level control, and there is an Initial Level control, just as we discussed above. A block diagram for all of this is shown in Figure 12, above.

As you can see, that's quite a lot of electronics. Of course, we shouldn't really call this unit 'just' a VCA, but synthesizer manufacturers often throw strict accuracy to the wind to make things easier to grasp (and thank heavens for that!). You may be curious as to why one of the CV inputs is called 'LIN' and the other 'LOG'. This leads us to a whole new chapter regarding the ways that signals exist and respond in the real world. Consequently, it too will have to wait till another time.

## ...And Finally

Before finishing, I want to leave you with one more thought. All through this article we've treated the main signal path as an audio signal path. But what's to stop us using CVs as the main input signals in Figures 4, 5, 6, 8, 9, 10 and 12? The answer is 'nothing', and this leads us directly to another Synth Secret:

*One of the most common uses for VCAs is to modify the actions of CVs using other CVs;*

...and, contrary to what I wrote in the second paragraph of this article, another:

*We should* not *differentiate between amplifiers in the audio signal chain and those used to modify control voltages.*

Of course, there's nothing to stop us passing the output from our first VCA to a second for further modification... and on, and on, and on. And, all the while, our first Synth Secret (suitably modified to take into account the actions of contour generators and other modifiers) still applies:

*Whenever you have more than one amplifier and/or attenuator in series, you can at any moment calculate the Gain of the whole system simply by multiplying the individual Gains — at each instant — together.*

This is one of the prime reasons why freely patchable (modular) synths are so powerful: you can dynamically modify any signal — audio or CV — using any other dynamic signal (or signals). And the devices that let you do this are... the lowly and often overlooked VCAs! SOS

**"This is one reason why modular synths are so powerful: you can dynamically modify any signal using any other signal. And the devices that let you do this are VCAs!"**

# synth secrets

## PART 10: MODULATION

In this month's instalment of his series on the basics of subtractive synthesis, Gordon Reid considers the magic ingredient that makes all the other elements sound interesting...

In the first nine episodes of Synth Secrets, we've covered a lot of ground. We have discussed waveforms, filters, envelopes, triggering and gating — surely we are now ready to create some killer synth sounds? No? What's missing? With the knowledge we now have, we can select our waveform, modify it using all manner of powerful filters, and apply envelopes to give the result life and dynamics. But this still isn't enough. Despite the contouring applied to the VCFs and VCAs, there is no movement within the sound: it still sounds like a 'beep' of some sort.

The missing element is modulation, the trick that makes sounds live and breathe in an organic sort of way. If you think that this simply means using an LFO to add a bit of vibrato, think again. Modulation is undoubtedly the most involved (and involving) subject in all of synthesis. Nevertheless, if we are going to graduate on to the more complex aspects of the subject, we're going to have to exercise some discipline. This means that, at first, we must confine ourselves to a discussion of the simplest forms of cyclic modulation.

### 1: Three Simple Modulations

Figure 1 combines the most important sonic elements from Parts 1 to 9 of this series, connected into a simple but usable synthesizer configuration. As you can see, the signal source is a tone generator of some sort (an oscillator). The output from this first passes through a filter, and then an amplifier before reaching the outside world. The contour generators 'shape' the sound by causing the filter and amplifier to affect the brightness and loudness of the sound at the second and third stages, respectively. (For clarity, I have shown audio signals with bold arrows, and control signals with lighter ones. For the same reason, I have also omitted the pitch CV, Triggers and Gates.)

Now consider someone playing a string instrument such as a violin, or a cello. As you know, the position at which the player presses the string on to the neck of the instrument determines the pitch of the note that's produced. The shorter the effective length of the string, the higher the pitch of the note. Now picture that person playing, and you will see that he (no sexism intended) often rocks his fingers backwards and forwards on the neck. This shortens and lengthens the string by a small amount each time he moves his hand (he is also changing the tension by a tiny amount, but we need not worry about that). This change, in the effective length raises and lowers the pitch slightly, resulting in vibrato. Figure 2 is a simplified representation of the way a short burst of vibrato modifies the pitch of an otherwise steady note. Likewise, Figure 3 shows — in grossly exaggerated form — the change in a waveform during a period of vibrato.

Reproducing this effect on an analogue synth, ▶

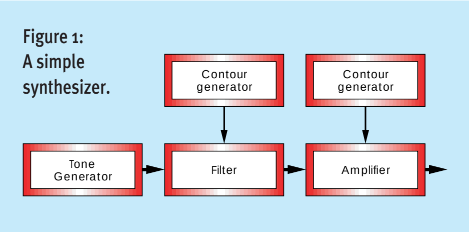


Figure 1: A simple synthesizer.

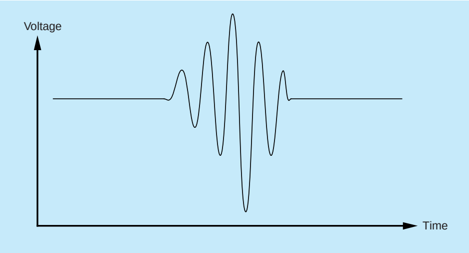


Figure 2: Applying this control voltage to the oscillator fequency would result in a short burst of vibrato.

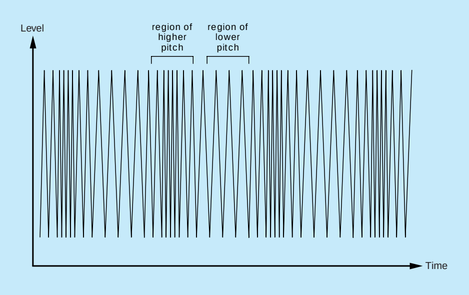


Figure 3: The effect of vibrato on a triangle wave.

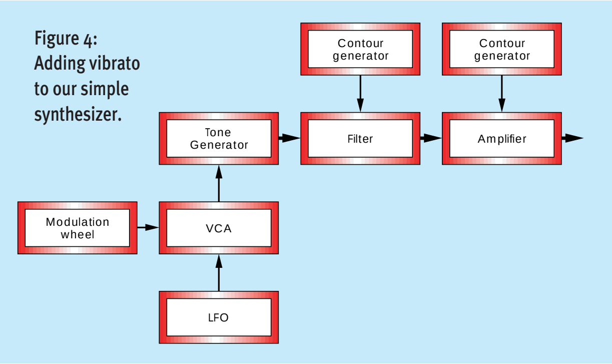


Figure 4: Adding vibrato to our simple synthesizer.

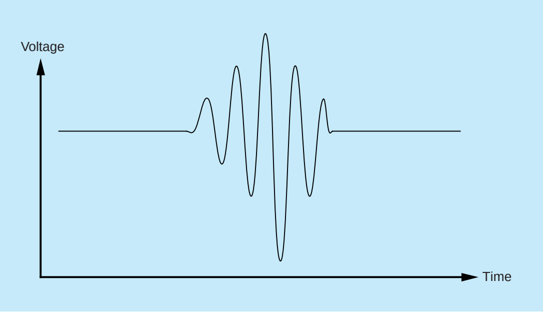


Figure 5: Applying this control voltage to the VCA would result in a short burst of tremolo.

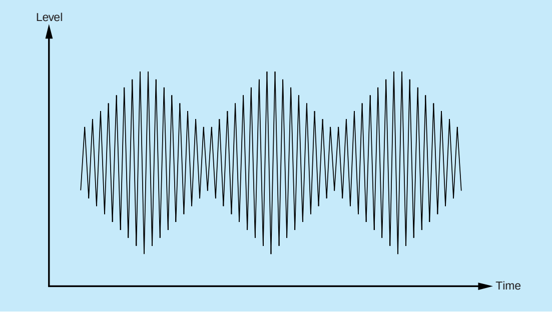


Figure 6: The effect of tremolo on a triangle wave.

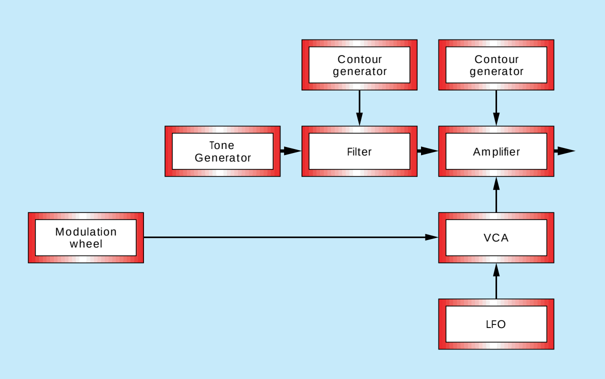


Figure 7: Adding tremolo to our simple synthesizer.

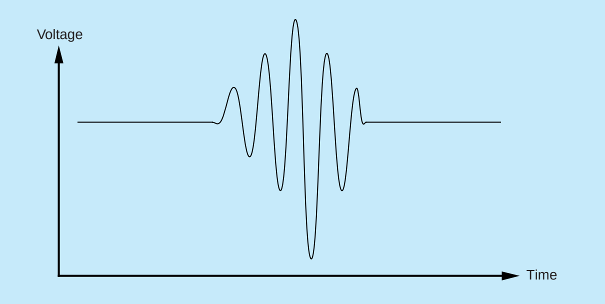


Figure 8: Applying this control voltage to the filter cutoff would result in a short burst of growl.

albeit in a rather unsubtle manner, is easy, and I am sure that most of you will be way ahead of me here. You simply apply a control voltage to the oscillator in such a way that it raises and lowers the pitch slightly.

If you look at the block diagram in Figure 4, you will find that the CV (the modulating waveform) is supplied by a low-frequency oscillator (an LFO). You will also notice that a VCA controls the amplitude of the modulating waveform, and the VCA is itself controlled by a modulation wheel. The VCA and wheel are very important parts of this configuration because, without them, there would be no simple way to control the amount of vibrato, which would be particularly unmusical.

## Tremolo

Let's now move on from vibrato, and consider tremolo. These effects are frequently mistaken for one another, but the distinction is simple: whereas vibrato is a periodic change in the pitch of a sound, tremolo is a periodic change in the loudness of the sound. Figure 6 has the same axes as Figure 3, but the graph has quite a different shape, showing how the frequency of the note remains constant, but the level changes in time.

Now that we understand the difference between these two effects, it is easy to see how the synthesizer differentiates between them. Whereas vibrato requires that something modulates the pitch of the oscillator, tremolo requires that something modulates the gain of the audio amplifier at the end of the signal chain. The block diagram for this looks like Figure 7.

The third in our introductory set of effects occurs when we modulate the voltage-controlled filter in our simple synthesizer. The results are, perhaps, less easy to envisage than either vibrato or tremolo. Indeed, you can obtain three distinct effects simply by changing the frequency of the LFO that generates the modulation in the sound. At the slowest settings (say, about 0.1Hz) you will obtain a slow filter sweep which is very useful for ambient sounds. If you increase the speed to around 1 or 2Hz this becomes more like a wah-wah effect. If you continue to increase the speed to the upper limit of subsonic oscillations (around 10 to 20Hz) you will hear a growl which can be superb for simulating brass instruments.

The effects of this type of modulation are less easy to draw (see Figure 9), because the visible changes in the waveform caused by modulated filtering are so slight. You can *hear* the changes very easily (this is not a subtle effect), but you would need an oscilloscope to see that the waveform is slightly rounded off when the cutoff frequency is low, and that the amplitude drops by a small amount as the upper harmonics are attenuated.

Of course, there is no reason why vibrato, tremolo and growl should be mutually exclusive, and a good analogue synthesizer will allow you to route the LFO to any of these destinations.
A better synth will offer individual LFOs that you can assign to each destination, with the depth of each effect governed by a selection of controllers. ▶

**"Pulse Width Modulation is ideal for creating string ensemble sounds, as well as rich lead synth patches."**

▶ These might include modulation wheels, aftertouch (pressure sensitivity), or foot pedals. The block diagram for this (Figure 11) may look daunting, but you will see that it breaks down into the easily understood parts shown above.

However, it's unlikely that you will want to control the level of each of the three modulation VCAs using all three controllers simultaneously. Therefore, you should consider the lines joining the controllers to the VCAs as options. This introduces us to the switches that select between sources and destinations on most synthesizers. The decisions you make regarding these will help determine the sound your synth produces, and the ways in which you can control the modulation to make your playing more expressive.

## 2: Another Simple Modulation

There is one more simple form of modulation that is implemented in most good analogue synthesizers. To understand this, let me take you back to a waveform I introduced when we discussed low-pass filters in Synth Secrets Part 4 (*SOS* August '99). This is the square wave, so-called not because its shape is truly square, but because it is a pulse waveform that spends the same amount of time at its peak as it does at its nadir (Figure 12).

Clearly, the square wave is a special case of the class of waveforms called pulse waves, all of which share the same 'rectangular' shape, but are different in as much as the ratio of the time the signal spends at the 'top' and 'bottom' of the wave differs. We call this ratio the 'duty cycle'. Since the square wave spends exactly half its time at the 'top' of the wave, it has a ratio of 1:2, and we say that it has a duty cycle of 50 percent. A similar wave that spends just one-third of its time 'at the top' has a duty cycle of 1:3 (33.3 percent), while one that spends a quarter of the time 'at the top' has a duty cycle of 1:4 (25 percent) — see Figures 13 and 14.

*Note: You will sometimes see references to duty cycles greater than 50 percent. In essence, for any number X lying between zero and 50, a duty cycle of (50+X) percent is identical to that of (50-X) percent, but the phase is inverted. For the purposes of this discussion, you need not worry about this.*

Pulse waves with different duty cycles have quite different audible characteristics. Narrow cycles (usually in the range 5 to 10 percent) are thin and nasal, and are often used to create sounds such as oboes. As the duty cycle becomes closer to 50 percent the sound thickens considerably, but at exactly 50 percent it has a distinctively hollow character that is ideal for simulating clarinets and other 'woody' sounds. (No sniggering from our American readers, please.)

These timbral changes are a consequence of the harmonics present in each waveform, and the amplitudes that they each possess. Fortunately, we can easily express the relationship between the duty cycle and the harmonic distribution, as follows:

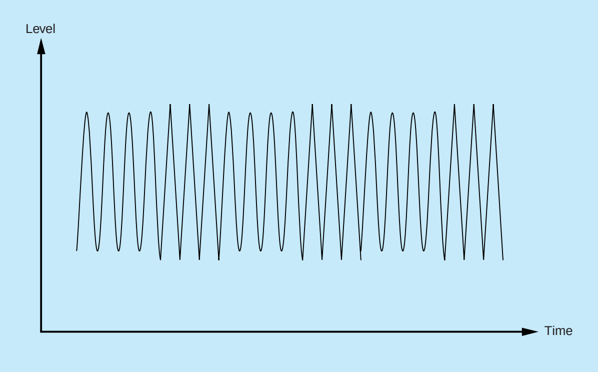


Figure 9: The effect of growl on a triangle wave.

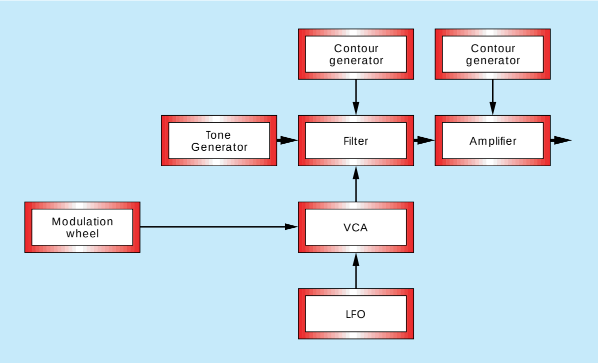


Figure 10: Adding filter sweeps, wah-wah, or growl to our simple synthesizer.

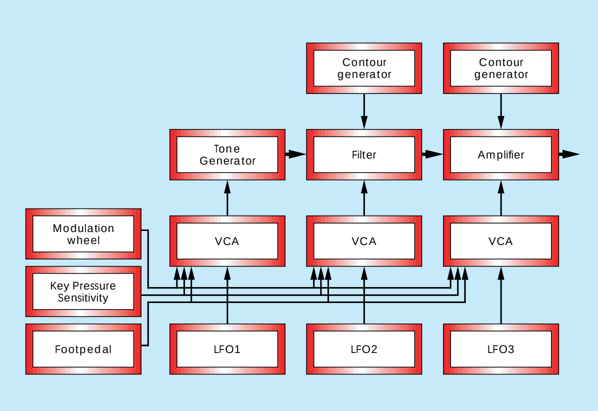


Figure 11: A simple synth will three LFOs and multiple modulation controllers.

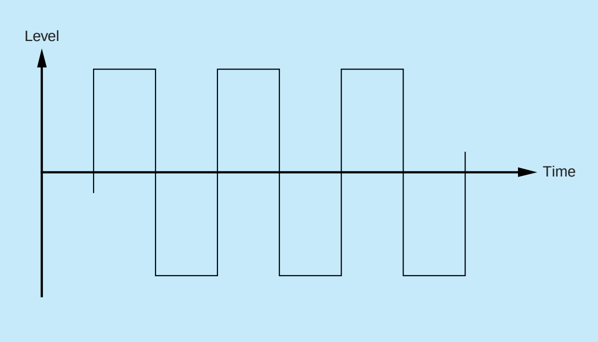


Figure 12: A square wave.

*A pulse wave has the same harmonic distribution as a sawtooth wave except that, for a duty cycle of 1:n (where n is an integer) every nth harmonic is missing from the spectrum.*

(If you need to remind yourself of the harmonic distribution of a sawtooth wave, please refer back to Synth Secrets 1, *SOS* May '99.)

So, armed with this knowledge, let's consider the square wave again. This has a duty cycle of 1:2, so we can infer that every second harmonic is ▶

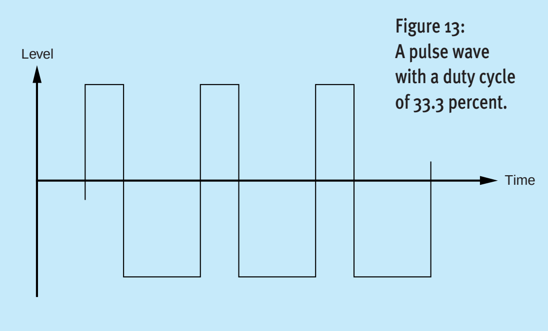


Figure 13: A pulse wave with a duty cycle of 33.3 percent.

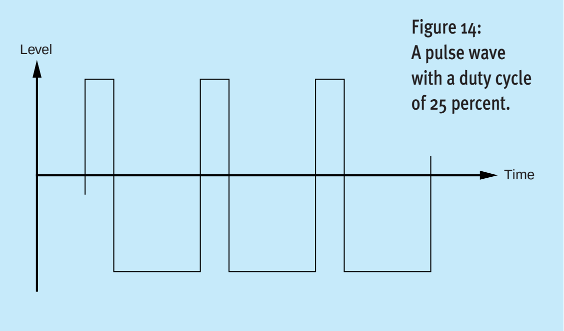


Figure 14: A pulse wave with a duty cycle of 25 percent.

missing from the spectrum. Our law describing the pulse-wave relationship therefore yields a result that people often quote, but rarely understand: since every even harmonic is missing, a square wave comprises the 1st, 3rd, 5th, 7th, and other odd-numbered harmonics.

Now consider the pulse waves in Figures 13 and 14. These have duty cycles of 1:3 and 1:4 respectively. Consequently, the 33.3 percent pulse wave includes the 1st, 2nd, 4th, 5th, 7th, 8th harmonics (and so on), with every third harmonic missing, and the 25 percent pulse wave includes the 1st, 2nd, 3rd, 5th, 6th, 7th, 9th, 10th harmonics (and so on), with every fourth harmonic missing.

You may ask what happens when the duty cycle is not an exact integer ratio. For example, what happens to the harmonics in a 28.5 percent pulse wave? The answer is intuitive. Because 28.5 percent lies somewhere between the 1:3 and 1:4 duty cycles, every third harmonic is somewhat attenuated, as is every fourth, but no harmonics are completely eliminated from the signal.

OK, are you completely happy with these concepts? Good, because now we're going to combine our knowledge of pulse widths and modulation to create... Pulse Width Modulation. This uses a second CV input to the oscillator which, instead of modulating the frequency, allows you to modulate the duty cycle of the pulse wave oscillation. If an LFO provides the modulating CV, the resulting waveform looks like Figure 15. The synth configuration that creates this is shown in Figure 16.

The configuration in Figure 16 is very common, and creates the lush 'chorused' sounds that make many analogue synthesizers so desirable. Although Pulse Width Modulation is usually applied to pulse waves, there are a handful of synths that allow you to apply it to sawtooth waves. Of course, you can't describe this in terms of duty cycles, and calling the modulation Pulse Width Modulation is somewhat misleading. Nevertheless, those synths that offer it (such as the Roland Alpha Junos) provide yet another range of subtly different timbres.

Pulse Width Modulation is ideal for creating string ensemble sounds, as well as rich lead synth patches. But be careful to avoid confusing it with the many chorus effects units that you can buy, or with the chorus effects built into some synthesizers. These devices split the final signal produced by the instrument into two parts, and then use delay and modulation processes to create their effects. Pulse Width Modulation is quite different, modulating the amplitudes of the individual harmonics at their source, and producing a unique range of sounds.

## 3: From Subsonic To Audio Frequencies

All the forms of modulation that we have discussed this month have one thing in common: they use low-frequency oscillators as the source of the modulating control voltages. This is perfectly valid, and the simplified synth configurations shown above provide most of the 'expression' in synth performances. But you may ask yourself what happens when we modulate the oscillator, filter and amplifier (or even the pulse width) using audio-frequency (AF) signals? At first, you might think that we would hear very rapid vibrato, tremolo, growl, or Pulse Width Modulation, but you would be wrong. Modulating audio frequencies with audio frequencies creates a completely different set of effects, some of which lead directly to a powerful form of digital synthesis. But that will have to wait until next month, when we'll really start to stretch some synth boundaries... SOS

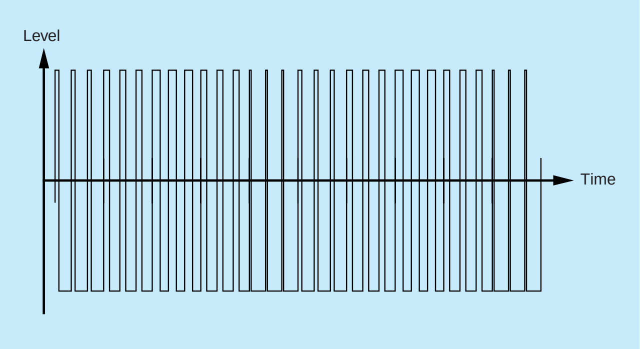


Figure 15: The waveform of a pulse width modulated pulse wave.

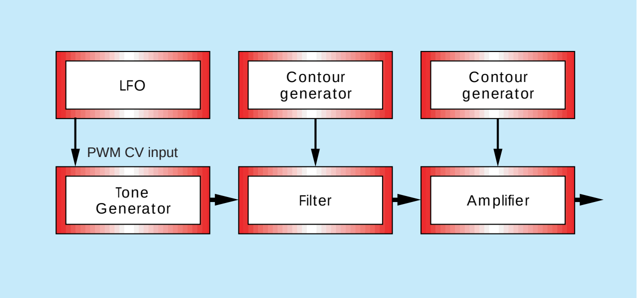


Figure 16: Modulating the pulse width using an LFO.

# synth secrets

## PART 11: AMPLITUDE MODULATION

Last month, **Gordon Reid** examined the concept of modulation at low frequencies. This month, he speeds things up a bit. The result is not just faster versions of the same modulation effects, but a new type of synthesis...

In Part 3 of this series, and then again last month at the end of Part 10, I introduced the idea that modulation sources do not need to be low-frequency oscillators or envelope generators. Of course, they frequently are, and most pre-patched analogue synthesizers are limited in this fashion. Fortunately, some powerful synths allow you to modulate their oscillators, filters and VCAs using higher-frequency sources whose output signals are within the range of human hearing (ie. audio-frequency signals). This opens the door to a whole new world of timbres that you can not easily obtain in any other fashion.

When the modulation source is an audio-frequency oscillator and the destination is the Gain of a VCA in the audio signal path, we call the result Amplitude Modulation, or AM for short. I find AM a fascinating topic, not least because it has a quite unexpected result: instead of just sounding like a very fast tremolo, it creates new frequencies that were not present in the original signals! But how does this happen? On a common-sense level, it's very counter-intuitive, so the answer, almost inevitably, lies in some maths. It's not exactly rocket science (to be precise, it's A-level trigonometry) but I will quite understand if you want to skip this section and jump directly to the examples in the second half of this article. But the more daring among you might find this interesting...

### Maths Can Be Fun... Honestly!

Equation 1 is the formula that relates the instantaneous amplitude (the level at any given point in time, called 'A') of a single frequency (called 'w') to time (called 't'). I could just as have easily written this using a sine term (it is, of course, a sine wave), but using a cosine changes nothing except the phase, and it makes the maths a little simpler. There is one other term in the equation, 'a', and this is the maximum amplitude of the waveform (the maximum level of the waveform in its cycle).

$$A_1 = a_1\cos(w_1 t)$$

Equation 1: A simple cosine wave.

You may also have noticed that every term in equation 1 (except time) has a small '1' subscript. This demonstrates that each part of the equation relates to our first waveform.

$$A_2 = a_2\cos(w_2 t)$$

Equation 2: A second cosine wave.

Now look at equation 2. This is identical to equation 1, except that all the subscripts are the number '2'. This shows that we have a second waveform to consider, and that this has a different maximum amplitude and a different frequency.

Let's consider the case where the amplitude $a_1$ of Signal 1 is half the amplitude $a_2$ of Signal 2. I will call these the Gains of 1 and 2 on some arbitrary scale. Now let's define $w_1$ to be 50 percent higher than w2 — say 300Hz for Signal 1 and 200Hz for Signal 2. I have plotted these waves in Figures 1 and 3 (right), and shown their harmonic spectra in Figures 2 and 4 (right). OK, so the equations may look a little arcane, and the graphs may look more familiar, but they are simply different ways of describing exactly the same information.

Now let's consider what happens when you mix these waves together. Figure 5 (see page 89) shows the synthesizer block diagram, Figure 6 (also page 89) shows the resulting waveform (which is just the arithmetic sum of the two waves at each moment), and Figure 7 (see page 90) shows the harmonic spectrum of the new waveform.

This is not a very interesting result. But now let's change our synthesizer configuration very slightly, and replace the mixer with a Voltage Controlled Amplifier (or VCA) in the signal chain. We will send Signal 1 to the audio input of the VCA, but instead of sending Signal 2 to another audio input we will use it

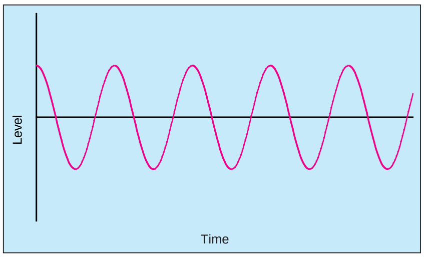


Figure 1. Signal 1: $a_1$=1, $w_1$=300Hz.

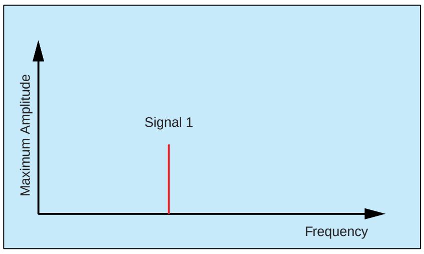


Figure 2. The harmonic spectrum of Signal 1.

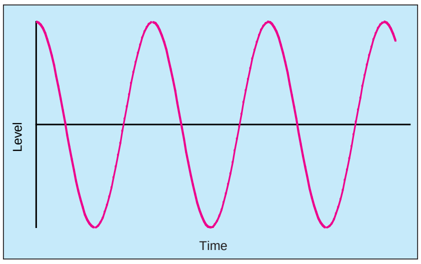


Figure 3. Signal 2: $a_1$=2, $w_1$=200Hz.

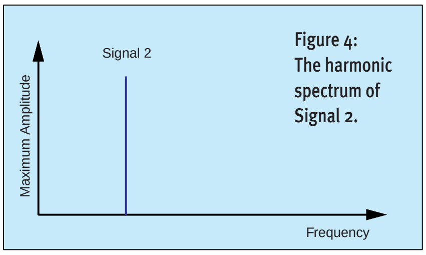


Figure 4: The harmonic spectrum of Signal 2.

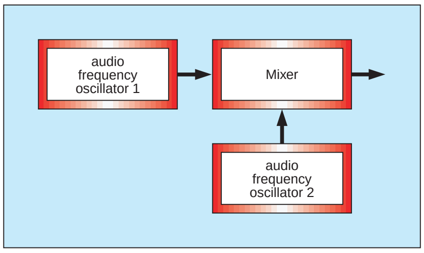


Figure 5: Mixing two audio signals.

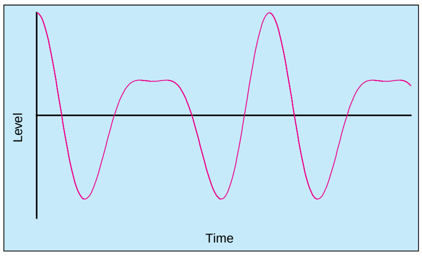


Figure 6: The result of mixing the two signals in Figures 1 and 3.

(like last month's LFO) to modulate the gain of the device. Signal 2 is, therefore, the Modulator, and Signal 1 must be the Carrier (See Figure 8, on page 90).

You'll remember that, in Equation 1, the term $a_1$ determined the maximum amplitude of the wave, and for the sake of argument I will define this as the Gain of the VCA. But we now have a situation where the Gain is being modulated by the instantaneous amplitude of the second signal. So, when the waveform of the Modulator is positive (ie. above the 0V axis) the Gain of the VCA increases, and when it is negative, the Gain of the VCA decreases. But, at every moment in time, we know exactly what the amplitude of the modulating signal is: it's $A_2$, as defined in Equation 2.

So we can now write a new equation, which shows that the output signal has an amplitude that is the sum of the original $a_1$ plus $A_2$. This is Equation 3.

$$A_1 = (a_1 + A_2)\cos(w_1 t)$$

Equation 3: The equation defining the output waveform.

Now, we know what $A_2$ is, so we can rewrite Equation 3 as Equation 4, and then rapidly rewrite *that* to obtain Equation 5.

$$A_1 = (a_1 + a_2\cos(w_2 t))\cos(w_1 t)$$

Equation 4: Another way of writing equation 3.

$$A_1 = a_1\cos(w_1 t) + a_2\cos(w_2 t)\cos(w_1 t)$$

Equation 5: Another way of writing equation 4.

This may look more complicated than the original Equation 3, but you can take my word that it expresses exactly the same information about the signals and the VCA. However, it's at this point that I'm going to ask you to take something on trust. If you look at the right-hand term in Equation 5 you will see that it contains two cosine terms multiplied together. This is a horrible thing to try to understand. Fortunately, there's a bit of maths that proves that you can separate a single term that multiplies two cosines of frequencies X and Y into two new terms of the form cos(X+Y) and cos(X-Y). If this sounds like complete gobbledegook to you, don't worry — accept that we can split the right-hand term in Equation 5 into two new terms, as shown in Equation 6. One of these is a wave of frequency $(w_1 + w_2)$, while the other is a wave of frequency $(w_1 - w_2)$.

$$A_1 = a_1\cos(w_1 t) + 1/2[a_2\cos(w_1+w_2)t] + 1/2[a_2\cos(w_1-w_2)t]$$

Equation 6: The result of Amplitude Modulation.

If you now look at Equation 6 more closely you'll see that the first term (immediately to the right of the 'equals' sign) is the original Signal 1 — in other words, the Carrier signal. Now consider what the frequency ($w_1 + w_2$) in the second term means... this must represent a wave with a frequency equal to the Carrier frequency plus the Modulator frequency. The third term must, therefore, be a wave with frequency equal to the Carrier frequency minus the Modulator frequency. In other words, Amplitude Modulation allows the original Carrier waveform through the VCA, and also creates two new signals called the Sum and the Difference signals. I love this stuff!

Now look at Figure 9 (right), which shows the waveform defined in Equation 6. As you can see, this signal is markedly more complex than the simpler mixed signal in Figure 6.

Moving on, Figure 10 (right) shows the spectrum of the waveform in Figure 9. Notice that the Modulator has completely disappeared, and that the Sum and Difference signals have half the amplitude of the Modulator (this is what the '1/2's in Equation 6 are telling us).

## In Use

OK, I'm sure that that's enough theory for this month, so let's ask ourselves why anyone would include AM capabilities in a synthesizer. To answer this, we'll consider two cases of Amplitude Modulation: one where the Modulator lies at a fixed frequency, and another where the Modulator as well as the Carrier tracks the keyboard (or any other controlling voltage).

Let's first ask what happens when you play the Carrier from your keyboard, but a Modulator (of equal amplitude) is fixed at a frequency of, say, 100Hz.

▪ CASE 1

When the Carrier also has a frequency of 100Hz (a moderately deep bass note), the three frequencies produced by Amplitude Modulation lie at 0Hz, 100Hz, and 200Hz. These are, of course, the Difference, the Carrier, and the Sum signals. You may be tempted to think that the signal at 0Hz has no effect, but this is not the case. It still has an amplitude (of half the Modulator amplitude) and this manifests itself as an offset in the signal. We call this a DC (direct current) offset because, being at 0Hz, it has no oscillation frequency. You can see this in Figure 11 (right) because a very high proportion of the output signal lies above the axis. DC offsets can have significant effects when a signal is processed by other synthesizer modules such as filters and amplifiers. Unfortunately, these effects lie outside the scope of this article...

The other two signals are the Carrier at 100Hz, and the Sum at 200Hz. The Sum signal is, of course, exactly an octave above the Carrier, so this will sound harmonic, or 'sweet'. We can represent this case by saying that the three components at the output lie at 0 percent, 100 percent and 200 percent of the Carrier frequency. So far, so good.

▪ CASE 2

Now let's play a few notes further up the keyboard, say at a frequency of 200Hz. This is our new Carrier frequency. Since the Modulator is unchanged at 100Hz the Difference signal now lies at 100Hz, and the Sum lies at 300Hz. The sound thus produced is still tonal to a degree because the Difference is exactly an octave below the Carrier. The Sum, however, is not harmonically related to the Carrier. Nevertheless, this example is another special case because the Sum is the third harmonic of the Difference, so the output still sounds 'musical', even though the Carrier (which is, in effect, the second harmonic of the Difference) is the dominant signal. In this case, we can represent the result by saying that the three components at the output lie at 50 percent, 100 percent and 150 percent of the Carrier frequency.

▪ CASE 3

Now let's choose a more random frequency for our carrier. What will happen if the Carrier frequency

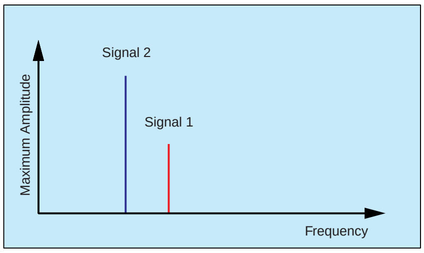


Figure 7: The harmonic spectrum of the mixed signals.

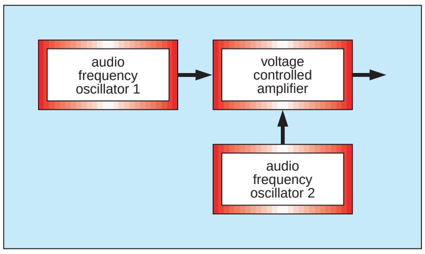

audio frequency oscillator 1
voltage controlled amplifier
audio frequency oscillator 2

Figure 8: Audio frequency amplitude modulation.

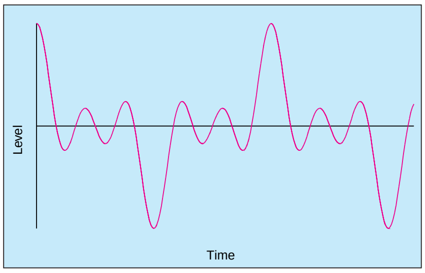


Figure 9: Amplitude Modulation of Signal 1 by Signal 2.

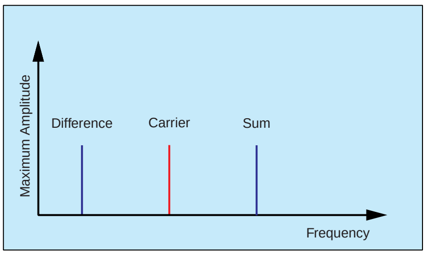


Figure 10: The harmonic spectrum of the waveform in Figure 9.

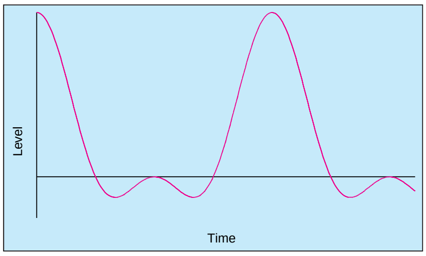


Figure 11: Amplitude Modulation of a Carrier with $a_1$=1 and $w_1$=100Hz by a Modulator of $a_2$=1 and $w_2$=100Hz.

is, say, 371Hz? The Difference and Sum signals now lie at 271Hz and 471Hz respectively, and there is no harmonic relationship between any of them. The three components at the output lie at roughly 73 percent, 100 percent and 127 percent of the Carrier frequency and the result is, therefore, enharmonic, and clangorous.

Indeed, enharmonicity is the result when virtually all signals are treated in this fashion. There are very few special instances such as Cases 1 and 2 above. Consequently, fixed-Modulator AM is most useful for creating aggressive and conventionally 'unmusical' sounds that change dramatically as you play up and down the keyboard. You can even control the amount of enharmonicity by raising or lowering the level of the Modulator.

But what happens when the Modulator is not fixed? How does this affect the sounds produced by your synthesizer? To answer this, let's consider the case where you're 'playing' both the Carrier and the Modulator using the same CV source to affect the frequency of both signals.

- CASE 4

As a starting point, we'll return to Case 2 above. The Carrier still has a frequency of 200Hz and the Modulator has a frequency of 100Hz. As before, the output contains the original 200Hz, plus the Difference and Sum signals at 100Hz and 300Hz. But this time, we're going to patch the synthesizer so that, as you play the Carrier up and down the keyboard, the Modulator frequency tracks the change in the Carrier frequency. So, for example, if you play a Carrier frequency of 400Hz (one octave higher than the initial 200Hz) the Modulator frequency doubles too, and the Difference and Sum frequencies become 200Hz and 600Hz respectively. In both cases, the relationship between the Difference, Carrier, and Sum signals is 50 percent, 100 percent and 150 percent. You can see this in Figures 14 and 15 (right): the frequency may have doubled, but the shape of the waveform itself has remained the same.

Indeed, no matter what initial frequencies you choose, the relationships between the Difference, Carrier and Sum remain constant if both the Carrier and the Modulator track the keyboard equally. As a result, you always obtain a consistent tone at the output. So this form of Amplitude Modulation offers a way to create complex non-harmonic timbres that change pitch normally as you play up and down the keyboard. To put it in a Synth Secrets sort of way:

*Amplitude Modulation is a powerful tool that allows you to create and play new sounds that you cannot obtain using conventional oscillators alone.*

This is true even though everything we have discussed assumes that we have used simple sine waves as our input signals. But you can, of course, use other waveforms. Imagine that each signal had a fundamental and one extra harmonic. Instead of two Sum and Difference signals, there would now be eight. If each signal had three components, the number of these 'side bands' would increase to 18. Four components would lead to 32 side bands... and so on.

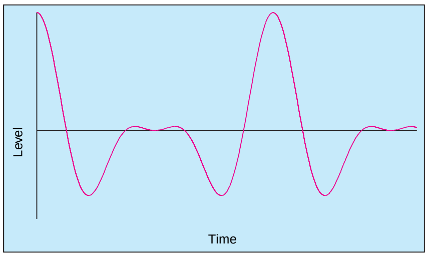


Figure 12: Amplitude Modulation of a Carrier with $a_1$=1 and $w_1$=200Hz by a Modulator of $a_2$=1 and $w_2$=100Hz.

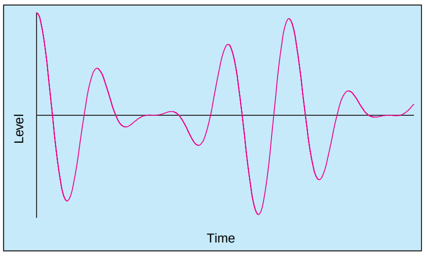


Figure 13: Amplitude Modulation of a Carrier with $a_1$=1 and $w_1$=371Hz by a Modulator of $a_2$=1 and $w_2$=100Hz.

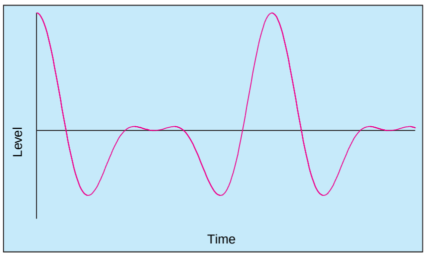


Figure 14: Amplitude Modulation of a Carrier with $a_1$=1 and $w_1$=200Hz by a Modulator of $a_2$=1 and $w_2$=100Hz.

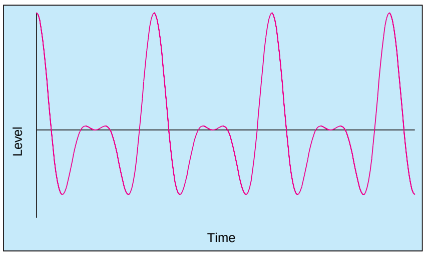


Figure 15: Amplitude Modulation of a Carrier with $a_1$=1 and $w_1$=400Hz by a Modulator of $a_2$=1 and $w_2$=200Hz.

Since there is nothing stopping you from modulating harmonically complex waveforms, why not use square waves and sawtooth waves as Carriers and Modulators? There's no reason why not. Fortunately, it's no harder to understand what happens with these waves than it is to understand simple sine waves, it's just more laborious...

Remember that a sawtooth wave contains all the harmonics in the conventional harmonic series. For example, a 100Hz sawtooth has components at 100Hz, 200Hz, 300Hz, 400Hz... and so on, while a 75Hz sawtooth has components at 75Hz, 150Hz, 225Hz... and so on. So what happens when you Amplitude Modulate a 100Hz sawtooth Carrier with a 75Hz sawtooth Modulator?

Unsurprisingly, the fundamental in the Carrier interacts with the fundamental of the Modulator, giving components at 25Hz, 100Hz, and 175Hz. It also reacts with the second harmonic of the Modulator, giving additional output components at -50Hz, 100Hz, and 250Hz (you might think that a frequency of minus 50Hz is a stupid concept, but in practice you hear this as simply 50Hz). Then there's the third harmonic of the Modulator (resulting in 125Hz, 100Hz, and 325Hz components) the fourth harmonic, the fifth, the sixth... ooh, this is complicated!

But before you think that you have grasped this and avoided a headache, don't forget that the second harmonic of the Carrier also interacts with the complete harmonic series of the Modulator, as does the third, the fourth, the fifth... and on and on and on and on! Fortunately, the amplitudes of all but the first few components are very small, so in practical terms you can discount the higher-order series. Nevertheless, it's not hard to imagine that these tones will be very complex. Indeed, they produce superb starting points for complex synthesis using filters and other modulators.

## Real VCAs & Ring Modulators

In all the above we have assumed that the VCAs in your synthesizer work perfectly (ie. according to signal-processing theory), and that none of the Modulating signal breaks through to the output. In my experience this is never the case, although with very good VCAs the leakage can be reasonably small. Nevertheless, even a small amount of Modulator will contaminate the output signal quite noticeably and, in general, this will increase the enharmonic quality of the result.

There is, however, another class of synthesizer modules that (when well designed and implemented) eliminate not only the Modulator but also the Carrier from the output. These are Ring Modulators, devices that have acquired a certain mystique over the past few years. Nevertheless, a Ring Modulator is merely a special case of an Amplitude Modulator. Furthermore, it only works in the fashion described when both the Carrier and the Modulator waveforms are precisely centred on zero volts. To facilitate this, many RMs are 'AC-coupled', which means (at least, in an ideal world) that any DC offsets in the inputs are eliminated before modulation. The result is an output consisting of the Sum and Difference frequencies, but neither of the input frequencies. Lesser RMs are 'DC-coupled' and respond somewhat differently from their AC-coupled cousins. Most notably, these allow the Carrier and Modulator signals to pass into the output. A few RMs, such as the device in the ARP 2600, have a switch that allows you to select between AC-coupled and DC-coupled modes — the best of both worlds.

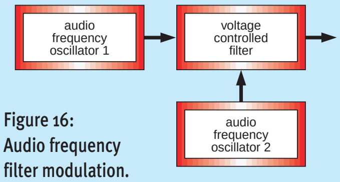

audio frequency oscillator 1
voltage controlled filter
audio frequency oscillator 2

Figure 16: Audio frequency filter modulation.

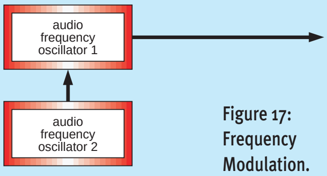


Figure 17: Frequency Modulation.

## Filter Modulation

To finish off, let's ask ourselves what would happen if we used Signal 2 to modulate the cutoff frequency of a low-pass filter rather than the gain of a VCA (see Figure 16, below left). You might fear that this would lead to another complex discussion with loads of new impenetrable mathematics. Happily, this is not the case.

Imagine a single harmonic of a complex waveform lying somewhere just above the cutoff frequency $F_C$. As you modulate $F_C$ you will find that sometimes the harmonic is attenuated more because of the modulation, while at other times it is attenuated less. In other words, this harmonic is being Amplitude Modulated by the changing action of the filter. Depending upon the width of the modulation (the maximum amplitude $a_2$ of the Modulator) the same is true to a greater or lesser extent for all the other harmonics within the signal. So this time, instead of having one set of harmonics modulating another set of harmonics, we have just a single set, but each component is being modulated in a different way. As I said before, I love this stuff!

## Frequency Modulation

OK, so that has explained Amplitude Modulation (which is tremolo at audio frequencies) and Filter Modulation (which is growl at audio frequencies). Surely, it's not too great a leap to describe what happens when we take vibrato into the audio frequency domain? We could call this Frequency Modulation (or 'FM' — see Figure 17, left) and it can't be too complex, can it?

Well, remember that I said that Filter Modulation did not entail a complex discussion with loads of new impenetrable mathematics? Unfortunately, that's exactly what FM does entail... see you next time!

# synth secrets

## PART 12: AN INTRODUCTION TO FREQUENCY MODULATION

As **Gordon Reid** explained last month, audio-frequency modulation of the amplitude of a signal can be a powerful synthesis tool. The possibilities expand still further when we consider what happens when you use one audio-frequency signal to modulate the *frequency* of another...

How many people can claim to have discovered an entirely new form of sound synthesis? John Chowning can. He did so by accident while experimenting with different types of vibrato at Stanford University in the mid-'60s. Chowning found that when the frequency of the modulating signal increased beyond a certain point, the vibrato effect disappeared from the modulated tone, and a complex new tone replaced the original. With hindsight, we can see that he had stumbled upon what is now the most common encoding technique used for public radio transmission (hence 'FM' radio). But what made his discovery so serendipitous was that unlike radio engineers, who work at very high frequencies, way above the limits of human hearing, Chowning was able to listen to the modulated waveform. He quickly discovered that FM is a very powerful method of synthesis and, in 1966, became the first person to compose and record a piece of music using FM as the exclusive means of sound generation.

Chowning and his associates spent the next few years refining FM, and laid down a sound mathematical and practical basis for the results they were achieving. Chowning then had the Stanford University Licensing Office approach a number of American manufacturers to see whether they would be interested in implementing it as a commercial method of synthesis. At a time when the Minimoog and ARP Odyssey ruled, polysynths were but a twinkle in electronic engineers' eyes, and 4-bit microprocessors were state-of-the-art devices, none of the American manufacturers saw the potential of FM. So it was almost in desperation that Stanford turned to Yamaha. An engineer named Mr. Ichimura was duly despatched to see Chowning, and the rest, as they say, is history.

As a direct consequence of Yamaha's amazing success throughout the '80s — the company sold millions of FM synthesizers, organs and home keyboards — we now think of FM as an exclusively digital process. But that is not the case. It is more practical to implement it in digital form, but the theory of FM is just as applicable to analogue oscillators, as we shall see...

### Once More, A Little Maths

Last month I explained Amplitude Modulation, and described some of the ways in which it allows you to create new sounds. Let's recap a little. Equations 1 and 2 show two instances of the simplest waveform: a sine wave. The instantaneous amplitudes of the waveforms (their levels at any given point in time, called 'A') are related to their gains (the maximum amplitude reached in their cycles, called 'a'), their frequencies ('w') and time ('t'). And, as before, the subscripts '1' and '2' denote waveform 1 and 2 respectively.

$$A_1 = a_1\cos(w_1t)$$

Equation 1: A simple 'cosine' wave.

$$A_2 = a_2\cos(w_2t)$$

Equation 2: A second 'cosine' wave.

If you refer back, you'll also remember that we defined the maximum amplitude of the first waveform as the gain of a VCA, and we modulated this using the second waveform, as shown in Figure 1. Equation 3 shows how I wrote the mathematics that describes this arrangement.

$$A_1 = (a_1+A_2)\cos(w_1t)$$

Equation 3: The equation defining the output waveform from the VCA in Figure 1.

But now we're going to mix things up a bit. Instead of modulating the amplitude of the first waveform, we're going to modulate its frequency. The block diagram that describes this is deceptively simple, as I have shown in Figure 2. Similarly, the equation describing Frequency Modulation (Equation 4) looks no more fearsome than that describing Amplitude Modulation. Indeed, if you look closely you can see that exactly the same terms are present in both, it's just that one of them ($A_2$) has changed position.

$$A_1 = a_1\cos((w_1+A_2)t)$$

Equation 4: The equation defining the output waveform from Oscillator 1 in Figure 2.

▶

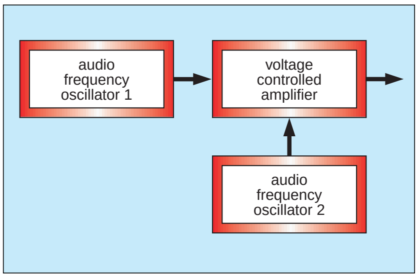

audio frequency oscillator 1
voltage controlled amplifier
audio frequency oscillator 2

Figure 1: Amplitude Modulation.

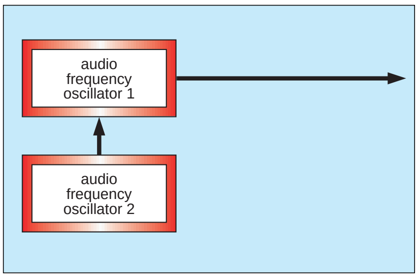


Figure 2: Frequency Modulation.

If we now echo last month's article and substitute the full expression for $A_2$ into Equation 4, we obtain Equation 5.

$$A_1 = a_1\cos((w_1+a_2\cos(w_2t))t)$$

Equation 5: Another way of writing Equation 4.

At this point, you would be completely justified in running, screaming, for the hills. Unlike last month's equations (which we could interpret using no more than 'O'-level or maybe 'A'-level maths) this one is a monster. Indeed, even with a degree in mathematics, you would be hard pressed to do much with it. You will, therefore, be delighted to know that I'm not even going to try to solve the equation. Unfortunately, that means that you will have to take many of the following facts on trust. But hey... trust me, I once lived with a doctor!

## FM Is Simply Very Fast Vibrato...

Now let's jump back two months, and return to part 10 of Synth Secrets and the simple vibrato that I described in that article. Figure 3 (right) shows what happens to a waveform (a 'Carrier') whose frequency is being swept up and down by a source of modulation (a 'Modulator'). In this example, the frequency of that Modulator is significantly lower than that of the Carrier.

Now let's ask ourselves what happens as we increase the Modulator's frequency until it approaches, equals, or even exceeds that of the Carrier. At some point, instead of looking like a cyclical 'squeezing' and 'stretching' of the Carrier waveform, the modulation will become a form of distortion within the individual cycles of the Carrier waveform. To demonstrate this I have drawn an example using a very short segment of Carrier waveform — maybe one-eighth of a cycle or thereabouts (Figure 4, right).

Let's now apply a Modulator to this. In this example it will have a low amplitude, but will be many times the frequency of the Carrier (see Figure 5) — since there are more than seven cycles of modulation in Figure 5, which shows one-eighth of a Carrier cycle, this means that the Modulator frequency is approximately 60 times that of the Carrier. As you will appreciate, this sounds nothing like vibrato. But what does it sound like?

## Side Bands, Side Bands Everywhere

If you refer back to Equation 5, you'll see that the equation for $A_1$ has two 'alien' terms within it: $a_2$ and $w_2$. These are, of course, the gain (maximum amplitude) and the frequency of the Modulator. So it's fair to assume that each of these will have an affect on the nature of the modulated signal. Let's look first at $w_2$, and see what attribute of the output is influenced by the Modulator's frequency.

John Chowning discovered that FM, like AM, generates side bands — additional components, not necessarily harmonically related to the frequency of the Carrier or Modulator — in the frequency spectrum of the output signal. (For an explanation of what side bands are, please refer back to last month.) To see how frequency modulation produces side bands, let's take an example of a sine wave Carrier with frequency $w_c$ and a sine wave Modulator of frequency $w_m$. I have shown these in Figure 6.

So far, so good... However, whereas AM generates just two side bands ($w_c+w_m$) and ($w_c-w_m$), FM produces a whole series that we can express as follows:

$$w_{sb} = w_c \pm n.w_m$$

Equation 6: The side-band frequencies, where $w_{sb}$ = the series of side-band frequencies, $w_c$ = Carrier frequency, $w_m$ = Modulator frequency, and n = any integer (0, 1, 2, 3, 4, and so on).

To put this in English: each side band lies at a frequency equal to the Carrier frequency **plus** or **minus** an integer multiple of the Modulator frequency. Of course, since 'n' can take *any* integer value, in theory, applying frequency modulation to a signal produces an infinite series of side bands. In the real world, however, no system has infinite bandwidth, and analogue systems are limited to producing side bands within their finite bandwidth (see page 90 for more on bandwidth). Similarly, manufacturers of digital FM systems constrain the mathematics to those values that they deem significant.

Fortunately, and despite this possible complication, the simple formula in Equation 6 makes it easy to see where the side bands are located. Given the Carrier and the Modulator shown in Figure 6, we can show the side bands as shown in Figure 7.

Now, what about the amplitudes of these side bands? OK, we now know that frequency modulation generates side bands, and that the Modulator's frequency determines where they lie. But what is the 'shape' of the resulting spectrum? To answer this, we must turn to the second attribute of the Modulator — its gain, $a_2$ — and introduce a new concept called the 'Modulation Index', or simply 'ß' ('beta').

To explain ß, I must again direct you to our example of simple vibrato. Imagine a simple synthesizer patch in which the amplitude of a modulation is modified by a VCA that is itself controlled by a Control Voltage source

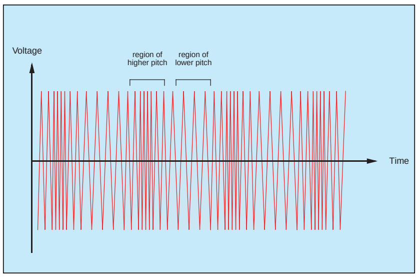


Figure 3: The effect of vibrato on a triangle wave.

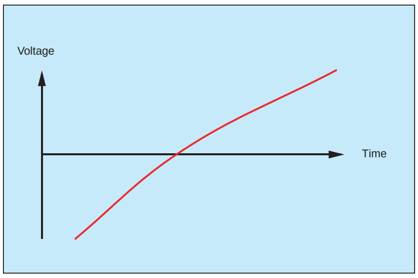


Figure 4: A short segment of a Carrier waveform.

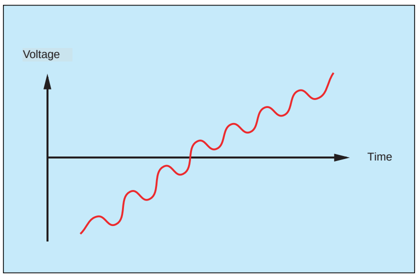


Figure 5: Modulating the Carrier in Figure 4 with a high-frequency, low-amplitude Modulator.

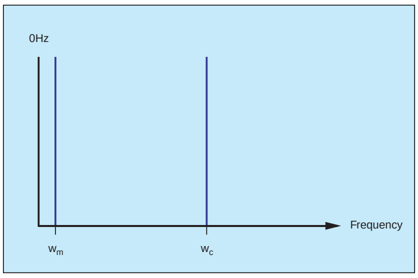


Figure 6: A Carrier and a Modulator.

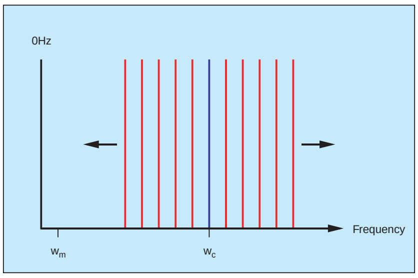


Figure 7: The positions of the side bands.

(see Figure 8). Don't forget that in this case, the Modulator frequency $w_m$ is very much lower than the Carrier frequency $w_c$.

Consider the case where the gain of the VCA is zero. Clearly, there will be no modulation, and the Carrier will produce a simple, unmodified tone. Now let's increase the gain of the VCA slightly. As you would expect, a gentle vibrato results, much like that of the aforementioned guitarist or violinist. But let's not stop here, and keep increasing the gain until the Modulator is sweeping the Carrier over a wide range of frequencies. At this point, a banshee wail or siren-type sound results. What we learn from this is that the sound we hear is not only determined by the frequency of the Modulator, but also by its gain or maximum amplitude.

Applying these ideas to audio-frequency FM, we must first define ß as the ratio of the Carrier's frequency sweep (the amount by which the Carrier deviates from its unmodulated frequency) divided by the Modulator frequency. We write this as follows:

$$ß = \frac{\Delta w_c}{w_m}$$

Equation 7: The Modulation Index, where ß = the Modulation Index, Δ (pronounced 'delta') means "the change in...", $w_c$ = Carrier frequency, and $w_m$ = Modulator frequency.

Since the numerator of this expression (the bit 'above the line') is the change in the Carrier frequency, this means that ß is directly related to the amplitude of the Modulator. Now, this is the point at which things get a little weird, because for any given Modulator frequency, it is the Modulation Index (and, therefore, the amplitude of the Modulator) that determines the amplitude of each of the components in the spectrum of the output signal. No, I can't demonstrate why this is so without invoking some of that scary maths I mentioned before, but I can show you a couple of examples.

Let's take that case where the Modulation Index is low — say in the region of 0.1 or less. The only significant side bands will be those closest to the Carrier frequency, and the result will look similar to that we obtained last month using Amplitude Modulation (see Figure 9, above). In contrast, if ß is significantly higher — say, in the region of 5 — we obtain a much broader series of side bands, and a much more complex spectrum results (see Figure 10).

I have shown the first six side bands created by ß=5, but there are far more in the real signal. What's more, you should note an interesting consequence of this value of ß: the amplitude of the original Carrier frequency has diminished significantly. Indeed, there is a value of ß that will cause it to disappear altogether!

Now look at Equation 7 again and you'll see that the denominator (the bit 'below the line') is the frequency of the Modulator. The consequences of this are very far-reaching. Let's say that you have decided that the spectrum in Figure 10 is the tone that you want, and that you want to be able to play the sound up and down the keyboard in conventional manner. This will require both the Carrier and the Modulator to track the keyboard equally so that the harmonic relationship between the spectral components (the side bands) remains constant. But Equation 7 demonstrates that, as the Modulator frequency increases, ß decreases. For example, if you play one octave higher, $w_m$ doubles and ß is therefore halved. To avoid this change in the spectrum, the Modulator amplitude must increase proportionally — it must double — to keep ß constant. This is not as much of a problem as it sounds, and Figure 11 shows the simplest way to achieve this.

Mind you, this configuration is almost impossibly difficult to calibrate perfectly, and the vagaries of analogue components ensure that it will, at best, be inconsistent. This is the reason why FM is almost always implemented using digital technology.

## FM Versus Analogue Filters

To move on, we need to talk about bandwidth for a moment. I first mentioned this concept in regard to Equation 6, when I said that real-world systems could not handle a signal of infinite bandwidth. So let's discuss what the real bandwidth of an FM'd signal might be.

For the purposes of this discussion, you could define the bandwidth as the range of frequencies occupied by any given signal. So, for example, a precise sine wave of, say, 100Hz would have negligible bandwidth (it exists only at a specific frequency) whereas a waveform with the same fundamental frequency plus one harmonic at 200Hz would occupy 100Hz of bandwidth. Likewise, a signal occupying the range between 100Hz and 1500Hz would have a bandwidth of 1400Hz, and so on. Now let's apply this concept to the output from an FM system.

Suppose that the Carrier is a sine wave of frequency 500Hz and the Modulator is a sine wave of frequency 300Hz. Clearly, if you mixed these together using a simple audio mixer, then using the simple definition above, the resulting signal would occupy a bandwidth of 200Hz.

Now let's configure the signals

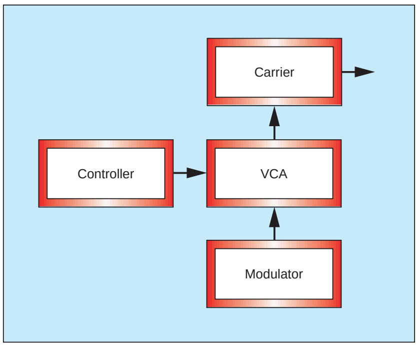


Figure 8: A simple analogue FM (vibrato) patch.

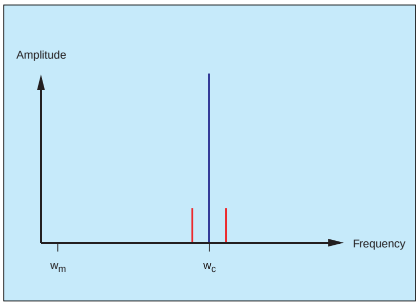


Figure 9: FM sidebands with low Modulation Index.

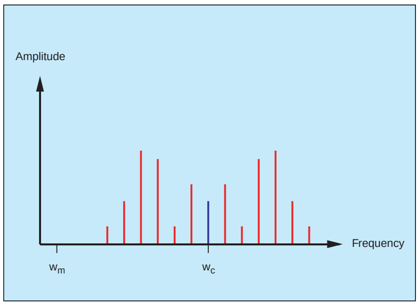


Figure 10: FM of the same signals when the Modulation Index is increased.

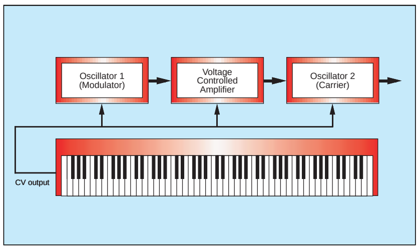


Figure 11: Keeping ß constant.

so that Amplitude Modulation occurs. From last month we know that the resulting three components have frequencies of $w_c$, $w_c+w_m$, and $w_c-w_m$. These frequencies are 500Hz, 200Hz, and 800Hz respectively, so the bandwidth of the resulting signal — the 'spread' between the lowest and highest side bands — is 600Hz.

Now let's consider the bandwidth of an FM'd signal. Although theoretically infinite (remember, the series of side bands is infinite) the Modulation Index will ensure that side bands of higher 'n' are of negligible amplitude. This means that the bandwidth is, to all intents and purposes, finite. Moreover, there is a 'rule-of-thumb' equation that gives us a rough idea of the meaningful bandwidth of the output signal. I have shown this in Equation 8.

$$B = 2\,w_m(1+ß)$$

Equation 8: The bandwidth of the modulated signal, where B = bandwidth, $w_m$ = modulator frequency, and ß = Modulation Index.

Let's say that ß is very small. Then, in our example of a 500Hz Carrier and a 300Hz Modulator, the bandwidth of the output will be equal to 2 x 300Hz x (1 + 0) = 600Hz. Thus, as I stated earlier, for low values of ß the result is much like that obtained using Amplitude Modulation.

But now let's suppose that ß=5. Then, in our example, the bandwidth of the output will be 2 x 300Hz x (1 + 5) = 3,600Hz. Clearly, high values of ß allow FM to create much more complex signals with a much higher bandwidth than the other methods of making two signals interact (see Figure 12). Note that this calculation also tells you that, in this example, there are 24 discrete spectral components in the output. Unfortunately, you'll have to read next month's Synth Secrets to find out why.

Now let's remind ourselves that, in simple subtractive synthesis, changes in the volume of a sound are most often determined by a contour generator acting upon an amplifier. Similarly, any changes in the tone of a sound are usually determined by a contour generator acting upon the cutoff frequency of a filter.

In an FM configuration, the volume of the sound is still determined by the volume envelope of the audio signal (which is, of course, the modulated Carrier), but you no longer need a filter to modify the tone. This is because, for any given Modulator frequency, the Modulator amplitude determines the bandwidth of the output. You can create an interesting demonstration of this using just seven synthesizer modules patched together as shown in Figure 13.

As you can see, the amplitude of the output signal will decrease as EG2 decreases the gain of VCA2 — so over time, the output gets quieter and quieter. At the same time, the maximum amplitude of the Modulator signal derived from VCO1 will increase as EG1 increases the gain of VCA1. This means that, as time passes, the Modulation Index increases, the bandwidth increases, and the output gets brighter and brighter. This is in marked contrast to natural sounds, where increased loudness almost always goes hand-in-hand with increased brightness.

You might think that there's nothing stopping you duplicating this effect using a filter, and it's true that you can use a contour generator and a low-pass VCF to brighten a signal as time passes. But that configuration would be quite incapable of recreating the complex tonal changes that also occur in the FM'd tone — changes that you cannot reproduce using conventional subtractive methods.

## Summary

So let's summarise. Without solving the mathematics of FM, we can say the following two things about the relationship between the Modulator and the output signal:

- The number of significant spectral components and their amplitudes are determined by the Modulation Index, which is proportional to the Modulator's amplitude; but inversely proportional to the Modulator's frequency...

...and...

- For any given Carrier frequency, the position of the spectral components is determined by the Modulator's frequency alone.

Although the proof of these statements and the calculation of the side bands' amplitude spectrum is a nightmare, the basics are easily understood. And, whether we perform FM using an analogue synth or a digital one, these principles remain the same. So here's this month's Synth Secret (which, as usual, is no secret at all):

*Frequency Modulation is a powerful method of synthesis that is as relevant to analogue synthesizers as it is to digital ones, and which is capable of generating sounds unobtainable by any other method.*

Next month, we'll take a closer look at some practical aspects of Frequency Modulation, and introduce the idea of 'operators' — a common concept if you own a DX7 synthesizer, but perhaps not one that you have encountered in analogue technology. We'll even take a look at some basic FM programming on a modular analogue synth.

Until then... SOS

> **"Frequency Modulation is a powerful method of synthesis that is as relevant to analogue synths as it is to digital ones, and which is capable of generating sounds unobtainable by any other method."**

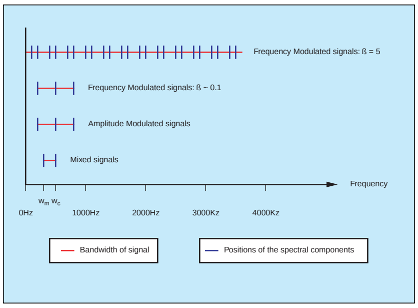


Figure 12: The bandwidths of mixed and modulated signals.

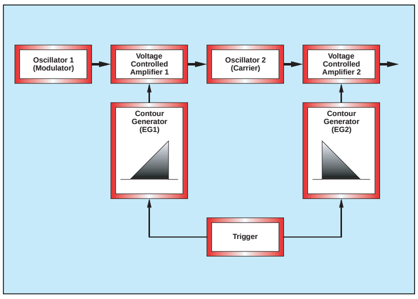


Figure 13: A simple analogue FM synthesizer.

# synth secrets

## PART 13: MORE ON FREQUENCY MODULATION

Last month, we examined the frankly scary maths allowing you to predict the audible effects of Frequency Modulation. This month, although the maths gets even tougher, **Gordon Reid** relates the theory to the practical implementation of FM synthesis on Yamaha's digital synths, as well as modular and non-modular analogues.

After reading last month's part of this series, you will (I hope) have progressed a fair way towards understanding Frequency Modulation. Nevertheless, if you were now presented with a digital FM instrument or a modular analogue synth patched to allow frequency modulation, the chances are that you would still make FM noises rather than musically pleasing timbres. This is because we have not yet discussed the nature of the relationship between the side bands (which are produced as a side-effect of the actual process of Frequency Modulation itself) and the Carrier (the signal being modulated). Nor have I explained how these relationships can be made to fit the simple harmonic theories explained right at the start of this series, which would allow you create tones that human ears would perceive as 'notes'. Furthermore, apart from a couple of basic diagrams, I have made no attempt to show you how to control the amplitudes of the side bands. So hang on to your hats, because this is where the fun (umm... headache?) really begins.

Yamaha's DX1 was the flagship of Yamaha's FM range of synths, offering digitally controlled 6-operator FM synthesis, a weighted keyboard, illuminated algorithm displays, and a price tag of just under £9500 on its launch! The much cheaper, and consequently more accessible DX7 also boasted 6-operator FM, but lacked a few of the bells and whistles such as the weighted keyboard. Nevertheless, it went on to become the synth sales success of the '80s.

### Introducing Carrier:Modulator (C:M) Ratios

We already know that each side band in a Frequency Modulated signal lies at a frequency equal to the Carrier frequency *plus* or *minus* an integer multiple of the Modulator frequency. I expressed this last month as follows:

$$w_{sb} = w_c \pm n.w_m$$

Equation 1: The maths for working out the frequencies of the side bands produced by frequency modulation.

In this equation, $w_{sb}$ = the series of side band frequencies, $w_c$ = the frequency of the Carrier, $w_m$ = the frequency of the modulator, and n = any integer (0, 1, 2, 3, 4...).

Now I'm going to risk complicating matters by eliminating references to frequencies in Hertz. We can express the relative frequencies of Carrier and Modulator using a ratio that I will refer to as the 'C:M Ratio'. For example, if the Carrier frequency $w_c$ is 100Hz and the Modulator frequency $w_m$ is 200Hz, I will refer to these as frequencies with a 1:2 ratio. I will then use C and M to refer to the Carrier and Modulator frequencies as they are expressed in the C:M ratio — what matters is the *relative* frequencies of Carrier and Modulator, not their absolute values in Hz.

Having made this definition, we can now say that, for any given Carrier frequency, the frequencies of the 'upper' side bands lie at C+M, C+2M, C+3M, C+4M... while those of the 'lower' side bands lie at C-M, C-2M, C-3M, C-4M... and so on. This is an important result, because it immediately associates itself with the idea of harmonic series.

Imagine an example where the C:M ratio is 1:1. In this case, the Carrier lies at '1', and the 'upper' side bands lie at '2', '3', '4'... while the 'lower' side bands lie at '0', '-1', '-2', '-3', '-4'... and so on. Now, if $w_c$ is

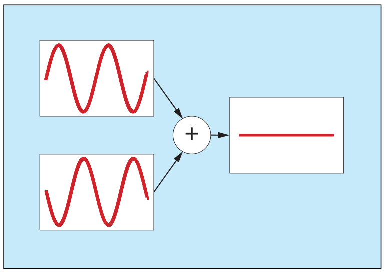


Figure 1: The cancellation of out-of-phase sine waves.

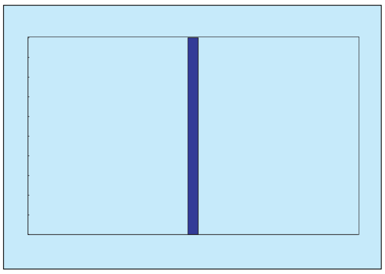


Figure 2: The amplitudes of the spectral components when ß=0.

Oberheim's Matrix 6 was a mid-'80s analogue synth with two digitally controlled oscillators per voice, which could be set up such that one was modulating the frequency of the other — in other words, all you need for a 2-operator FM synth. The Matrix 6's bigger brother, the *überanalogue* Matrix 12, can also be used in this way, as can tOberheim's Xpander rack synth.

100Hz, then $w_m$ must also be 100Hz (it is a 1:1 ratio), and the upper side bands must therefore lie at 200Hz, 300Hz, 400Hz... *ad infinitum*. This, of course, is a perfect harmonic series for a 100Hz wave. In other words, the upper side bands of a 1:1 C:M ratio produce a harmonic series no matter what the Carrier frequency may be.

But what about the lower side bands? If we extend our argument, these lie at 0Hz, -100Hz, -200Hz, -300Hz... also *ad infinitum*. Hang on... what are negative frequencies?

It turns out that there is a simple answer to this. Negative frequencies are the same as positive frequencies, but with their phases inverted. Yet, as we showed in Part 4 of this series, out-of-phase signals will cancel out, leaving silence (see Figure 1, left). So surely the 1:1 ratio will result in silence as the 100Hz component cancels the -100Hz component, 200Hz cancels -200Hz ... and so on?

The answer, fortunately, is 'no', and the reason for this is simple: for total cancellation to occur, the amplitudes of the cancelling components must be equal, and in this case they are not. So how do we know what the amplitudes of the side bands will be?

## Enter The Bessel Functions

If you've been reading this series from the start, you'll have noticed that the term 'Bessel Function' has cropped up several times. This is because Bessel functions are a family of equations that describe some fundamental characteristics of the universe in which we find ourselves. Just as the value of Pi (3.141592654... and so on and so on) is much more than simply something to do with the circumferences of circles, Bessel functions crop up in almost every aspect of maths, physics and engineering. What's more, they are the key to FM because, if you know the value of the Modulation Index (as discussed last month), you can calculate the amplitude of any spectral component using a Bessel Function.

Now, if you want to skip the next few paragraphs, I wouldn't blame you in the slightest, particularly if you're not a fan of maths. What's more, leaving this next bit out won't interfere with your understanding of what comes afterwards. But the more adventurous among you may like to read on...

Let's define a couple of things that will make life simpler. Let's call the Carrier 'C' the 'zeroth-order' components of our modulated signal, 'C+M' and 'C-M' the first-order components, 'C+2M' and 'C-2M' the second-order components... and so on.

Having done this, I can tell you (but not prove without recourse to a serious maths text book) that the amplitude of each pair of side bands of order 'n' (remember, n is any whole number from 0 to infinity) is defined by a Bessel function of order 'n'. I have shown this in Equation 2. ▶

$$J_{(n)}(ß) = \sum_{k=0}^{\infty} \frac{-1^{k} * \left(\frac{ß}{2}\right)^{(n+2k)}}{k! * (n+k)!}$$

Equation 2: The mathematical nightmare that is a Bessel Function!

In *this* equation, J(n)(ß) is the nth-order Bessel Function for any Modulation Index ß, k is just an integer 0, 1, 2, 3, 4... up to infinity, ! is the algebraic term for 'factorial' (if you don't understand this, don't worry about it), and ∑ means 'the sum of all the terms from k=0 to k=∞.

Go on, admit it... this is your worst nightmare. What does this have to do with synthesis and making music? Well, quite a lot, actually. Every time you program a DX synthesizer you're using this. It's just that Yamaha have (kindly) hidden it from view. More importantly, every time you use the 'Cross Modulation' function on analogue synths that are lucky enough to have one, the nasty noises you create are determined by this equation — cross modulation being, of course, just another name for frequency modulation.

Anyway, if you'll bear with me for a little longer, I'll show you that Equation 2 isn't actually *quite* as bad as it looks. Imagine the modulation index ß is 0.1, and you want to calculate the amplitude of the zeroth component (ie. the Carrier). We can calculate the first term (when k=0) as shown in equation 3, and the second term (k=1) as shown in equation 4.

$$J_{(0)}(ß=0.1) = \sum_{k=0} \frac{-1^{0} * \left(\frac{0.1}{2}\right)^{(0)}}{0! * (0)!} = \frac{1}{1} = 1$$

Equation 3: The first term in calculating the amplitude of the Carrier.

$$J_{(0)}(ß=0.1) = \sum_{k=1} \frac{-1^{1} * \left(\frac{0.1}{2}\right)^{(2)}}{1! * (1)!} = \frac{-0.0025}{2} = -0.00125$$

Equation 4: The second term in calculating the amplitude of the Carrier.

You now do this for all the remaining terms (k=2, k=3, k=4... and so on) and then add them all up. If you could be bothered to do this, you would find that the third term is even smaller than the second (and positive), the fourth term is even smaller than that and negative, and so on. This called a converging series, and if you add up all of its terms, it demonstrates that the amplitude of the Carrier remains very close to 1, its initial amplitude. In principle, this calculation should take an infinite amount of time because k represents every whole number from 0 to infinity — but fortunately, anything beyond the third term is so tiny that it is irrelevant.

OK, so that's the amplitude of the Carrier. Now, what of the side bands? To work out the amplitudes of the first order side bands C+M and C-M, you just substitute n=1 in Equation 2, reset k to zero, and start the whole process again. When you have done this, you substitute n=2 in Equation 2, reset k to zero again, and calculate the amplitudes of the second-order side bands C+2M and C-2M. Then you substitute n=3, n=4... and so on, and on, and on... As you can see, you have an infinite series of infinite series to calculate, so I'll keep a table reserved for you at the Restaurant At The End Of The Universe. (Unfortunately, you'll be late.) Oh yes, and when you've finished that, Wiggins Minor, you can start on ß=0.5 for me. Argghh!!!

Fortunately, there are powerful numeric methods that will calculate all these amplitudes for you. Even the common (but very powerful) Microsoft *Excel* will give you the values of any Bessel function of order n and Modulation Index ß. I have, therefore, created a simple spreadsheet to generate the amplitude charts, shown as Figures 2 (on page 78) and 3-7 (right). These include the Carrier (the zeroth order component, in blue) and the first eight components (in green) on either side. As you can see from Figure 7, the amplitudes of the Carrier and first-order side bands are negative when ß=5. This does not mean that these frequencies are, by some mathematical sorcery, transported into a negative universe, it means that they are present in the output but with inverse phase.

## Creating Recognisable Harmonic Series

Let's now return to our example of a 1:1 series with ß=1. Looking at Figure 5 you can see that the Carrier retains much of its original amplitude, and that — looking to its right — there is a harmonic series at 2C, 3C, 4C and 5C. Now, looking to the left, we see that there is a significant component (C-M) at 0Hz (known as a DC or direct current component, because it has no oscillation frequency, as discussed in Part 11 of this series), and low-amplitude components at -C, -2C and -3C. These, as already discussed, reflect with inverse phase to C, 2C and 3C, and are therefore subtracted from the in-phase components at the same frequencies — see Figure 8 (on page 82). If we ignore the DC component, the resulting spectrum contains all the overtones of the Carrier frequency (1:1 is the only ratio that does this) and it looks like a 1/n harmonic series with just the first few harmonics present. If you remember the first part of this series, you'll recall that this describes a filtered sawtooth wave perfectly! And sure enough, that's exactly how it sounds.

Now let's consider the case of a 1:2 frequency modulation with ß=1. In this case the upper side bands exist at C, 3C, 5C, 7C... and so on, while the subtractive negative components reflect back at C, 3C, 5C, 7C... and so on. So, in this case, the result is a truncated harmonic series with

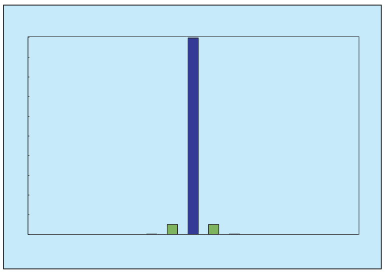


Figure 3: The amplitudes of the spectral components when ß=0.1.

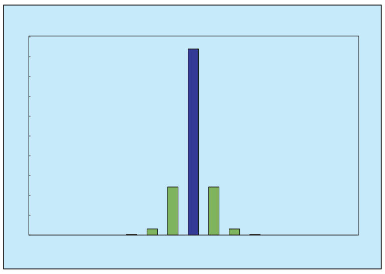


Figure 4: The amplitudes of the spectral components when ß=0.5.

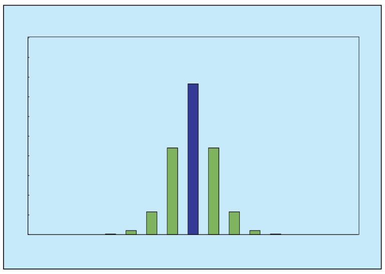


Figure 5: The amplitudes of the spectral components when ß=1.

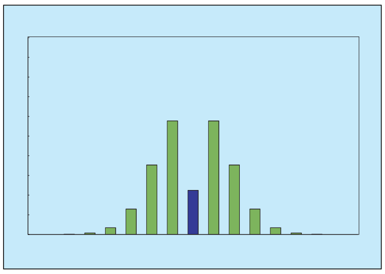


Figure 6: The amplitudes of the spectral components when ß=2.

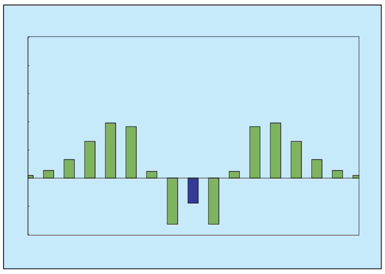


Figure 7: The amplitudes of the spectral components when ß=5.

just the odd harmonics present. Does this seem familiar? It should, because it's what you get when you filter a square wave! And, once again, that's how it sounds.

Moving on, what about C:M ratios of 1:3, 1:4, and other integers? Given what we've already discussed, you don't need diagrams to work these out. The ratio 1:3 gives upper side bands at 4C, 7C, 10C... and reflected lower side bands at 2C, 5C, 8C... which (ignoring phase inversion) is reminiscent of the spectrum of a 33% pulse wave. Similarly, 1:4 produces upper side bands at 5C, 9C, 13C... and reflected lower side bands at 3C, 7C, 11C... which is again similar to that of a square wave.

By the way, this explains why there were 24 spectral components in last month's example of an output with 3600Hz bandwidth. There were 10 upper side bands in the signal, the Carrier, one unreflected lower side band, and 12 reflected lower side bands.

At this point, you might be thinking that there's nothing too complicated about this FM malarkey (apart from maybe the maths), but like most things in life, it's not as straightforward as it might appear. The integer C:M ratios are relatively straightforward because they produce harmonic waveforms with the fundamental (which, by the way, may not be the loudest component) at the Carrier frequency. But these are special cases, and there is an infinite range of non-integer ratios from which to choose. I'll leave it to you to work out the spectrum of, for example, a 1:1.2138754 C:M ratio but, clearly, the resulting sound will be completely enharmonic. Not only that, but the Carrier will no longer be the lowest frequency in the spectrum.

## FM Synthesis On Analogue Synthesizers

Some analogue polysynths such as the Oberheim Matrix 6 are capable of simple, 2-oscillator FM. This is because their digitally controlled oscillators (DCOs) are stable enough to maintain the precise frequency ratios required for the technique (see last month's instalment of Synth Secrets for more on this). However, FM is only truly versatile when you have access to a large number of oscillators, VCAs and contour generators. Furthermore, however many modules you need for a sound, you will need twice as many for duophony, three times as many for 3-note polyphony... This soon becomes a very large number of modules, and is undoubtedly the reason why FM never caught on in the analogue realm. Just consider the size of the modular synth that you would need to emulate a DX7: 96 oscillators, 96 VCAs and 96 multi-stage EGs for the oscillator levels, plus pitch envelope generators and their associated VCAs, innumerable mixers, and heaven-knows-what-else. The total system would incorporate hundreds of modules and would weigh many tons.

Despite this, you can still produce interesting monophonic FM sounds using just a handful of analogue modules. What's more, these need not be part of a modular synth; they could just as easily be the sections within a pre-patched monosynth.

Take a look at Figure 9 (right). This shows a basic 2-oscillator FM configuration that uses just seven modules. Now we'll assign some numbers to the modules. Let's tune both OSC1 and OSC2 to the same frequency for a 1:1 ratio. Now set the amount of EG1 to zero (so that it is 'off'), and define EG2 to be a 0/0/10/0 (no attack, no decay, full sustain, instantaneous release) organ-type envelope. If you play a note, you will hear something that sounds similar to a sawtooth wave, as described above. Furthermore, you can fine-tune the pitch and the timbre by adjusting the gain of VCA1.

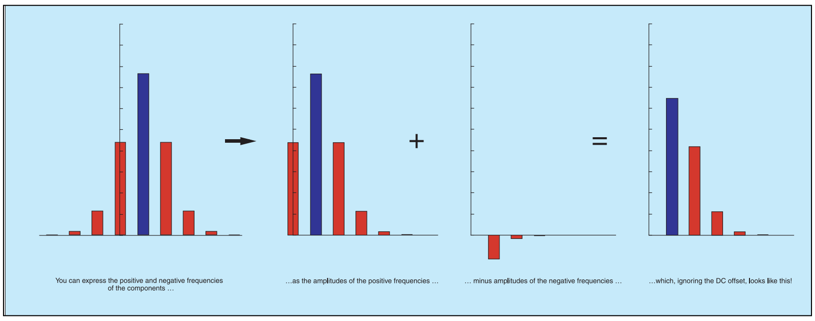


Figure 8: The spectrum of a 1:1 frequency modulation with ß=1.

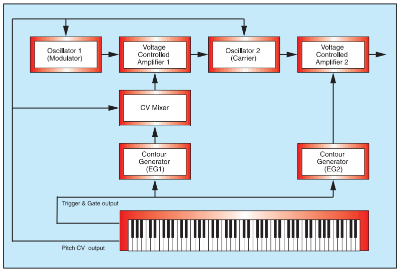


Figure 9: Simple FM on a modular analogue synth.

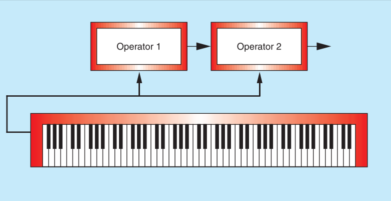


Figure 10: Another way of drawing Figure 9... this time as a '2-operator' FM synthesizer.

Having done this, you can now play with the gain of VCA1 and apply contours from EG1 to create radical timbral changes that often sound like nothing so much as sci-fi sound effects. If you now repeat this exercise with OSC2 set to twice the frequency of OSC1, your basic waveform will now sound more 'hollow', and much like a square wave.

Of course, you can easily let your imagination run riot, and use all manner of unrelated frequencies as starting points, or use scaled CVs to make the oscillators track at different rates. These will give you wildly varying sounds, all of which will be far more complex that those created simply by mixing the outputs of any two analogue oscillators. Indeed, with appropriate enveloping of the Carrier output, these sounds can be excellent for creating drums and other powerful percussion effects.

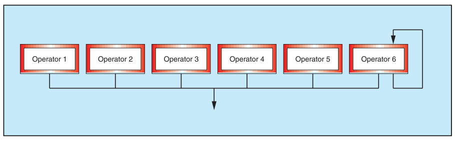


Figure 11: 6-operator FM: algorithm 32 from a Yamaha DX7.

## Operators

Unfortunately, block diagrams such as Figure 9 become very complex very quickly, so we need a shorthand way to represent them. We do this by introducing the concept of an operator — a combination of an oscillator plus any associated envelope generators, mixers and VCAs. Once defined in this way, we can redraw Figure 9 as Figure 10 (right) — which is much easier to understand. Having done this, we can create all manner of routings (or 'algorithms') in which operators affect each other in different ways. One of

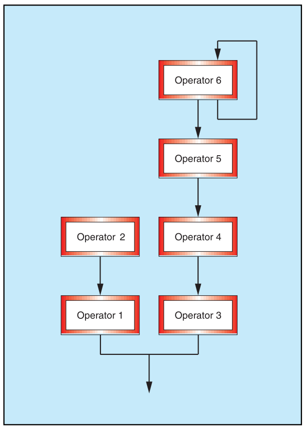


Figure 12: 6-operator FM: algorithm 1 from a Yamaha DX7.

the simplest examples is the 6-operator algorithm in Figure 11 (see page 82). This is often called the 'organ' on a DX7 because the outputs from each of the operators are summed together, with no FM taking place whatsoever.

In contrast, Figure 12 (page 82) shows a more complex arrangement. If you're familiar with Yamaha's 1980s digital implementation of FM (as seen on their DX synths, to name just one series), you should recognise such diagrams, as each digital Yamaha FM keyboard synth had all its possible algorithms printed on its top panel. Reading these algorithm diagrams is easy, as long as you remember that the operators acting as Carriers are always on the bottom row of the diagrams, with the Modulators arranged above them.

In Figure 12, therefore, Operator 2 is acting as Modulator for Operator 1 (a Carrier) while, at the same time, Operator 6 is modulating Operator 5 which is modulating Operator 4, which is modulating Operator 3, which is another Carrier. The outputs from the two Carriers are then summed together using a simpler mixer, and you hear the combined results of the two signal-generation paths. The benefits of this arrangement are, I hope, obvious. For example, you can create an evolving sustained tone using Operators 1 and 2, and use Operators 3, 4, 5 and 6 to add a complex 'attack' partial at the start, thus giving the sound life and character.

Figure 12 raises two important further points. The first is that you can cascade Operators, so there is nothing stopping you from modulating Modulators. The second is that you can mix the outputs from Carriers, as they are just conventional audio signals — so you can treat them as partials within the sound.

There are two other configurations you can use as building blocks in your FM patches, and I have shown these in Figures 13 and 14 (right). The first is that of two Modulators modulating a single Carrier. You might think that this would lead to unpredictable results, but in practice you obtain the arithmetic sum of the two predicted waveforms. To make that statement clearer: if Operator 2 (a Modulator) and Operator 1 (the Carrier) would in isolation produce a square wave, and Operator 3 (another Modulator) and the Carrier would produce a sawtooth wave, the output from Figure 13 would look like a square wave and a sawtooth mixed together.

The second configuration is that of one Modulator simultaneously affecting two Carriers (see Figure 14). In this case Operator 3 (the Modulator) acts independently on Operators 1 and 2 (the Carriers) in just the way it would if it were modulating only one of them. The results of these are added together by a simple audio mixer.

## Feedback

You'll have noticed that some of the operators in these diagrams show outputs that loop back on themselves. These, for obvious reasons, are called feedback loops, and they dramatically change the nature of the operator. Let's consider the operator in Figure 15 (see right), and say that this is producing a sine wave of 100Hz. Then, following our reasoning above, it is also receiving a 100Hz sine wave (its own output) as a Modulator, thus making it produce a complete harmonic series at its output. You can then use an input level control or a VCA within the feedback loop to control the brightness of the output waveform. Neat, huh?

Finally, let's prove that FM synthesis is not just the preserve of the DX-series Yamaha synths. Figure 16 (below, right) shows a dual-operator feedback configuration that doesn't exist within Yamaha's FM system, but is simple to patch on a modular analogue synth.

As you can see, there are two operators, and the output of the second, modulated operator is fed back to the input of the first operator. If the frequencies of the two operators are identical, this is no different from the configuration shown in Figure 15. But what happens if their frequencies are different and not related by integers? Now you have trouble! Look at it this way... the sine wave output by Operator 1 produces FM side bands in the output of Operator 2. This complex spectrum is then fed back to Operator 1, producing a hugely complex spectrum that is itself then fed to Operator 2, further complicating the output spectrum. This complex spectrum is then fed back to Operator 1, producing a hugely complex spectrum that is itself then fed to Operator 2, further complicating the output spectrum... As you might imagine, the output soon includes thousands — even millions — of frequencies, and deteriorates almost instantly into noise. This makes analogue FM an unlikely, but hugely powerful generator of drum and percussion sounds (refer back to Part 2 of this series if I've lost you).

## Conclusion

For some people, the last two months of this series may have looked like a DX7 tutorial, and it's true that everything I have written (except for the last example above) is applicable to this instrument, plus the dozens of other digital FM synthesizers produced by Yamaha, Korg and Elka. But that's to miss the point... everything I have written is applicable to *any* synthesizer that lets you direct the output of one oscillator to the pitch CV input of another. That includes the ARP 2600, the Minimoog (switch VCO3 to modulate VCO1 at audio frequencies), and the EMS VCS3, not to mention many other monosynths, DCO-based polysynths with 'cross modulation' or FM capabilities and, of course, all freely patchable modular synths.

So don't ignore FM — it's a powerful part of analogue synthesis. What's more, it produces sounds that conventional forms of subtractive synthesis cannot, filling holes in your sonic armoury that you never knew you had. SOS

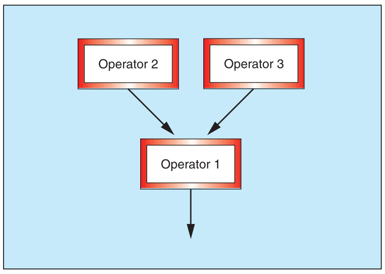


**Figure 13:** Multiple Operators affecting a single Carrier.

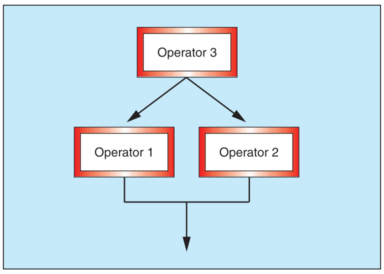


**Figure 14:** A single Operator affecting multiple Carriers.

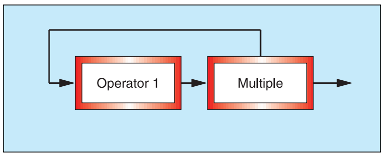


**Figure 15:** Feedback in an FM system turns a sine wave generator into a sawtooth generator.

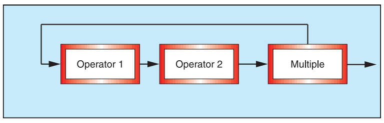


**Figure 16:** Feedback in a 2-operator FM system turns two sine wave generators into a noise source.

# synth secrets

## PART 14: AN INTRODUCTION TO ADDITIVE SYNTHESIS

Every pitched sound can be thought of as a collection of individual sine waves at frequencies related to the fundamental. **Gordon Reid** introduces a powerful method of synthesis that works by manipulating these individual harmonics.

For the past two months Synth Secrets has concentrated on frequency modulation, showing (I hope) that FM synthesis (or 'Cross Modulation' as it often used to be called) is as relevant to analogue synthesizers as it is to the digital synths that made it a household name. So now it's time to move on — to another realm of sound creation that is normally associated only with digital synths. This month's Synth Secrets takes us into the murky world of Additive Synthesis.

### The Principle Of Additive Synthesis

The concept underlying additive synthesis is very simple, and I can best explain it by turning all the way back to Synth Secrets Part 1 (*Sound On Sound* May '99). In this, I showed that you could represent any waveform as a set of sine waves. For a simple harmonic oscillator, each of these sine waves has a frequency that is an integer multiple of the fundamental frequency, and we call these the 'harmonics' of the sound. Just to refresh our memory, let's take the most common synthesizer waveform — the sawtooth wave — as an example.

Figure 1 shows an idealised sawtooth wave. You'll never see this in nature because the universe doesn't allow physical objects such as air molecules or the cones of a 4x12 cabinet to accelerate or move infinitely quickly. Unfortunately, this is what the ideal waveform requires as it moves instantaneously from its nadir to its zenith, but we're not going to worry about that. Now, you may recall that this waveform has a simple harmonic relationship, expressed as follows: every harmonic is present, and the amplitude of the nth harmonic is 1/n times that of the fundamental. We draw this as shown in Figure 2.

It's important that you fully appreciate that, within limits, Figures 1 and 2 represent exactly the same thing. I have truncated the number of harmonics in Figure 2 to just nine whereas there should, in theory, be an infinite series, but neither my screen nor your copy of *SOS* is infinitely wide, so this will have to do. If you're worried that truncating the series so severely will ruin my argument, take a look at Figure 3. This is the waveform generated by the nine harmonics in Figure 2, and no others. It's remarkably close to the ideal sawtooth, don't you think?

Moving on, let's ask ourselves the following question. If we can represent a waveform at any given moment by describing its harmonic content at that moment, is it reasonable to assume that we can take a harmonic series and derive a unique waveform from it? Of course it is! (Well, to be precise... as long as we overlook the phase relationships of the harmonics, of course it is!) This is because, as I have already stated, the waveform and the harmonic series are simply different ways of expressing the same thing.

Armed with this knowledge, we can take our series of nine harmonics and use them to create a huge range of waveforms. For example, let's give each of the nine the same amplitude. If we now assume that these are the only harmonics within the sound, we can calculate the waveform. (See Figures 4 and 5.) As you can see, this waveform looks quite unlike the one we had before. OK, so there's a passing resemblance, but all the 'squiggles' in the wave show that it has a much greater high-frequency content, and indeed it sounds very much brighter than the sawtooth of Figure 3. Likewise, you could generate a passable approximation to a square wave by using your knowledge of which harmonics are present and in which quantities. (See Figures 6 and 7.)

So here is the basis of additive synthesis: because, at any given moment, you can describe any waveform in terms of the frequencies and amplitudes of its components, you can take the appropriate number of sine waves and mix them together at the appropriate frequencies and in the appropriate quantities to regenerate

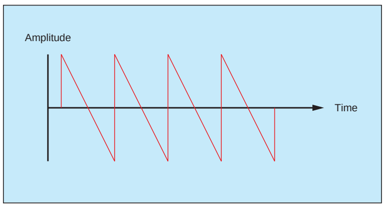


Figure 1: An idealised sawtooth waveform.

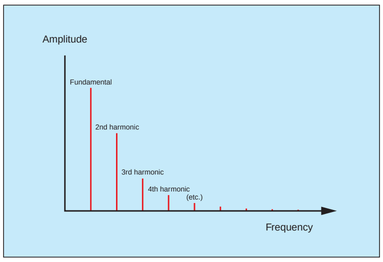


Figure 2: The first nine components in the harmonic series of a sawtooth wave.

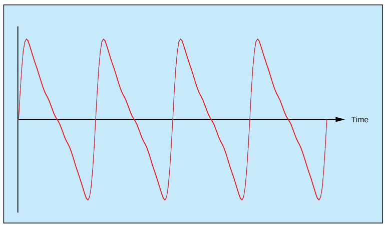


Figure 3: The waveform generated by the nine harmonics in a 1/n series.

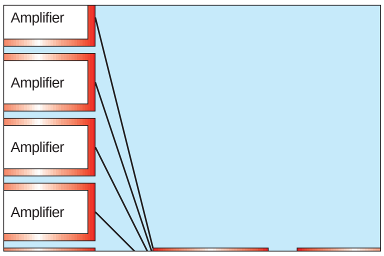


Figure 4: The nine harmonics of a waveform, each with equal amplitude.

the waveform. Indeed, if you have a large modular synth you can easily recreate the examples shown above. All you need are nine oscillators, nine VCAs, a mixer, and some form of Gate pulse to open the amplifiers when desired (see Figure 8). Yes, it's hopelessly inefficient, but the principle holds.

## An Early Electronic Analogue Synthesizer

If, in the analogue domain, additive synthesis were limited to monstrously over-endowed modular synths, you might think that this would be the end of our story. But it isn't, so this isn't. The choice of nine harmonics in each of these examples is not an accident, because it describes a very common analogue, additive synth. Sure, you may not think of it in this way, and you may be surprised to discover that it predates what we now think of as 'conventional' VCO-VCF-VCA analogue synthesis by about 30 years. This instrument is the Hammond Tonewheel Additive Synthesizer. Oops, sorry. I mean, it's the Hammond Organ.

Provided that the Hammond in question is a classic 'drawbar' model, not one of the modern ones with 'tabs' for the sounds, let's dispel any doubts that you may have regarding its status as a powerful additive synthesizer. To do this, I'll describe the Hammond itself in a bit more detail...

The sound of a tonewheel organ is generated by 91 discs sitting on an axle that runs much of the length of the instrument. Each of these is shaped like an old thre'penny bit so that, when rotated in front of a pickup, it generates an electrical current that is pretty close to sinusoidal (ie. pretty close to a sine wave). If you have just one drawbar extended when you play, each key taps the output from just one disc, thus making each note a reasonably pure sine wave. (This statement ignores the distortions introduced by the valve circuitry that infests a vintage Hammond, but we're

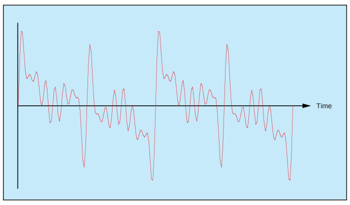


Figure 5: The waveform generated by the harmonic series in Figure 4.

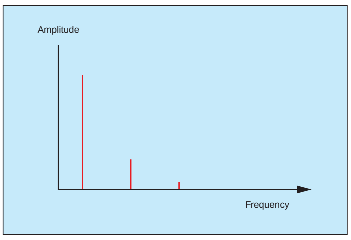


Figure 6: The first three harmonics of a square wave.

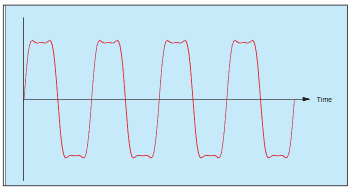


Figure 7: The waveform produced by just the first three 'square wave' harmonics.

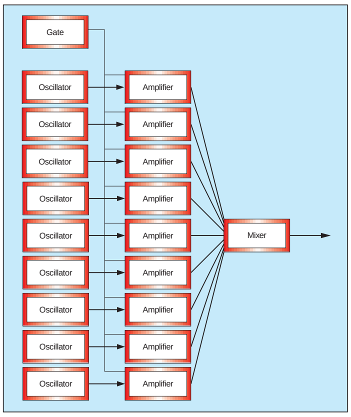
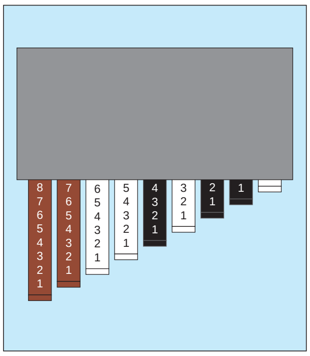


Figure 8: Configuring 20 modules as an additive synthesizer.

not considering such delicacies here!) If you extend a second drawbar simultaneously, you will add the output from another disc into the sound. This means that you will now have two sine waves per note. Pull out a third, and a third sine wave is added... and so on.

Figure 9 shows the classic Hammond configuration consisting of nine drawbars (see table). Each of these has nine amplitude positions (1 to 8, plus 'off') so many millions of possible combinations (more usually called 'registrations') are available. (There are a handful of Hammonds with more than nine drawbars per registration, and the spinet models have just seven on the lower manual, but we're going to ignore complications such as these.)

So there you have it: nine harmonically related pitches, each with nine possible volumes, and you can combine these in any way you choose. It's a very small leap to realise that this is, almost by definition, an additive synthesizer capable of producing millions of unique waveforms. But surely this can't be the be-all and end-all of additive synthesis? When all is said and done, the Hammond sounds, well, like an organ, not a powerful synthesizer. There's obviously something missing.



| Footage | Colour | Name | Pitch | Relationships |
|---|---|---|---|---|
| 16 | Brown | Bourdon | Sub-Fundamental | Sub-Fundamental |
| 5 1/3 | Brown | Quint | 5th | Sub-3rd Harmonic |
| 8 | White | Principal | Fundamental | Fundamental |
| 4 | White | Octave | 8th | 2nd Harmonic |
| 2 2/3 | Black | Nazard | 12th | 3rd Harmonic |
| 2 | White | Block Flote | 15th | 4th Harmonic |
| 1 3/5 | Black | Tierce | 17th | 5th Harmonic |
| 1 1/3 | Black | Larigot | 19th | 6th Harmonic |
| 1 | White | Sifflote | 22nd | 8th Harmonic |

Figure 9: The classic Hammond configuration of nine drawbars per registration.

## An Analogue Additive Synthesizer

We started this article recapping Synth Secrets 1, so now let's jump forward a few months to Synth Secrets 4 to 8. In my discussions about filters and envelopes, I postulated that sounds will always sound static and uninteresting if they do not change in time. So this gives us a clue to today's problem: the Hammond, while a powerful signal generator, has no means to shape or contour those signals into something more involving. So let's encapsulate this in another Synth Secret:

*Organs sound like organs not because of the simplicity (or not) of their waveform generators, but because their sounds do not change over time.*

Or, to put this another way:

*No matter how clever the method of synthesis, and no matter how complex an initial waveform may be, any timbre will sound static and 'organ-like' if it does not change in time.*

One way to add interest is by applying 'effects' such as phasers, flangers, or echo units to the basic signal. Unfortunately, these do not affect the essential nature of the sound. Indeed, the Hammond has its own particular set of effects — chorus/vibrato, reverb, and the wonderful Leslie rotary speaker — and these help to give the instrument its distinctive sound. But you could not call these effects a method of synthesis, so we must look elsewhere if we are significantly to improve our additive synthesizer.

Consider sampling the output from a tonewheel Hammond without the chorus/vibrato, reverb, or Leslie effects. Now consider playing this sample through the contoured filters and amplifiers that no doubt reside within your sampler. As you might imagine, the result would sound much more like a conventional synthesizer, albeit one with a more complex initial waveform than that produced by conventional oscillators. This then suggests how we can modify the 'instrument' in Figure 8 to design a more interesting additive synthesizer: simply add a time-varying filter and a time-varying amplifier after the output from the mixer (Figure 10).

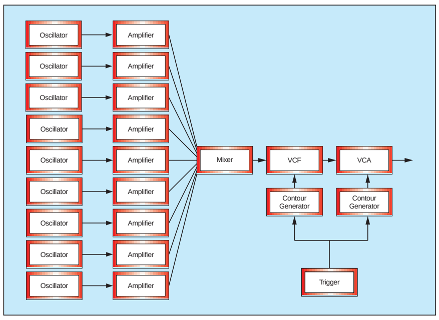


Figure 10: Adding filter and amplitude contours to our basic additive synthesizer.

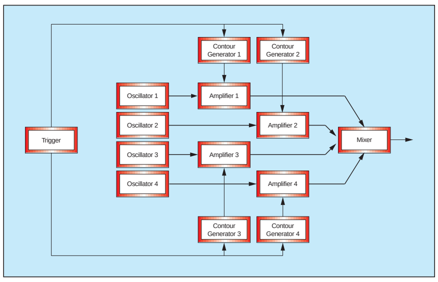


Figure 11: A more complex and useful additive synthesizer.

However, this still is not a very interesting additive synthesizer. Indeed, if we ignore the absence of modulators (and the fact that this discussion has limited itself, so far, to sine wave oscillators) this is not much different from a multi-oscillator synth such as a Minimoog. It's just that we have nine oscillators instead of three.

Now, consider the evolution of a real sound such as a plucked string. We know from experience that this is loud and bright at the start of the note, and becomes quieter and 'darker' as time passes. So let's take this simplistic description, and see how we can modify the 'synth'

### Additive Synthesizers

**I first experienced additive synthesis in the late '70s during a brief encounter with a Fairlight CMI. This was a dream machine, and I fell in love with the concept of being able to manipulate the very building blocks of a sound. In the mid-'80s Kawai released the K5. With its powerful additive engine it was, in theory, capable of all manner of sounds inaccessible from conventional analogue or digital synths. Unfortunately, the reality did not live up to the promise and the K5, while interesting, suffered from the bane of many '80s digital synths: unless treated with a great deal of love and attention it sounded sterile and uninvolving.**

**The final stage in this tale of additive lust brings us to the present day and my Kawai K5000S, a synth I like so much that I recently asked my producer Nick Magnus to buy one so that I didn't have to move mine between our studios. Ahhh... a happy ending!**

in Figure 10 to recreate these tonal changes more accurately. Firstly, we must assign the pitches of the oscillators to imitate the harmonic nature of the string. This is simple — it's the 1/n harmonic series discussed many times before. Secondly, we must consider how each of these harmonics changes in time. This is also simple: we know that the sound becomes duller as time passes, so the higher-frequency harmonics must decay more rapidly than the lower ones. Thirdly, we must determine how the overall brightness and loudness of the sound changes as the note progresses, and create filter and amplifier profiles that emulate this. But hang on a minute — if a sound can, at any instant, be determined by the relative pitches and amplitudes of its constituent harmonics, we have no need for these filters and amplifiers — the changes in the harmonics do all the work for us. This understanding then leads us to Figure 11, which is much closer to describing how a real additive synthesizer works.

As you can see, this instrument lacks the filters and output VCA of a conventional synth. However, it is still capable of creating most of the timbres of a typical VCO-VCF-VCA configuration, plus many, many others besides.

So let's now design our simple plucked string sound. For example, let's say that Oscillator 1 produces a sine wave at the fundamental (1st harmonic) frequency, Oscillator 2 produces a sine wave at the 2nd harmonic frequency, and so on. Now, let say that Amplifier 1 causes the sound of Oscillator 1 to decay from its full level to silence in some time T, Amplifier 2 causes the sound of Oscillator 2 to decay from its maximum level to silence in half the time, T/2... and so on. These relationships mean that the higher harmonics are louder at the start, so this sound is particularly bright in the first instance, much like a plucked or hammered string. Note also that, because the higher frequencies are decaying more quickly, the sound becomes 'darker' as time passes. This is akin to a low-pass filter following a simple AD contour with A=0 and D=T. I have shown in Figure 12 the four envelopes produced by the four contour generators.

If we now compute the waveform, we can see that the high frequencies decay quickly and that, by the time that the waveform decays to silence, only the fundamental remains. I have shown the individual sine waves in Figure 13 and the combined output in Figure 14.

This result is much as you would expect, although in all fairness you could more easily produce it using a sawtooth oscillator and a low-pass filter controlled by a single 2-stage contour generator. But now let's ask ourselves what happens if we make individual harmonics change in less obvious ways. How about making the third and fourth harmonics start quietly, get louder and louder, and then decay quickly to zero at the end of the note? Now we have a situation where the four contours are as shown in Figure 15, and the output waveform looks like Figure 16.

Looking at Figure 16 you can see that the waveform becomes much more complex as time passes. If you are experienced in looking at such waveforms, you can also see that the high-frequency content starts to dominate about half way through the note. This is a result that you simply can't obtain on a Minimoog, Odyssey, Prophet 5, or any other synthesizer with a single signal path. OK, you can approximate this simple example on synths with multiple signal paths (such as the Korg 800DV and, at the other end of the spectrum, the Prophet 10), but even these are limited to the simplest of such cases. In contrast, a true additive synthesizer will allow you to manipulate individually the amplitudes of 32, 64, 128, or even 256 harmonics, and that's something that no pre-patched analogue synthesizer can do.

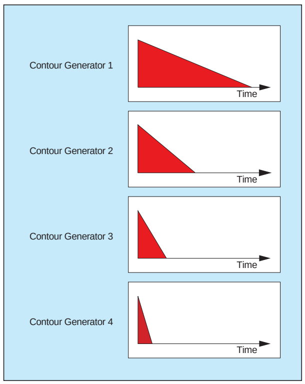


Figure 12: The four contours controlling the amplifiers in Figure 11.

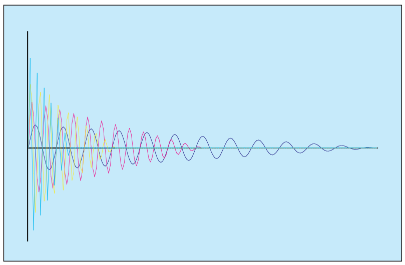


Figure 13: The four decaying harmonics defined by the contours in Figure 12.

## Fourier Synthesis And Beyond

This method of generating a complex sound is often called Fourier synthesis. (This is in honour of Joseph Fourier, the mathematician who discovered the basis of what we now call Fourier analysis — the mathematical method used to break sounds down into sine waves — and Fourier synthesis — building them back up again.) However, the more general term 'additive synthesis' does not presuppose that your oscillators are limited to sine waves. There is nothing stopping you from using square waves, sawtooth waves, or more complex waves such as PWM'd pulse waves or the outputs of ring modulators to create extremely complex time-varying spectra. Nevertheless, these complex waves can themselves be broken down into their constituent sine waves, so the underlying principle is always the same.

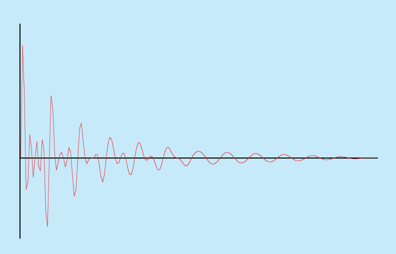


Figure 14: The final waveform output by our 'additive' synthesizer.

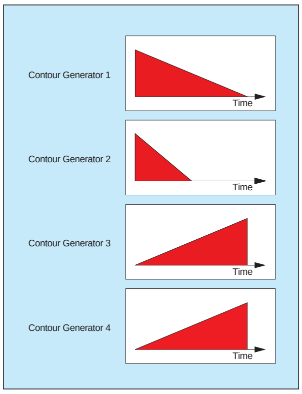


Figure 15: A combination of harmonic contours that you cannot obtain on a conventional analogue synthesizer.

Unfortunately, if you're after complex, evolving, and involving sounds, you're going to need a lot of sine-wave oscillators in your additive synthesizer. This is why the technique is always implemented in digital technology rather than analogue. After all, in the digital realm the oscillators are merely numbers in an equation, whereas the analogue additive synthesizer will require tons (literally) of VCOs, EGs, VCAs, and mixers. And, while there's nothing stopping you from creating additive sounds using just a handful of oscillators, you're going to need dozens or even hundreds if you're going to try to recreate natural sounds that contain a lot of overtones.

But even this isn't the end of the story because, for realism, each oscillator will require modifiers that modulate its pitch and amplitude. Without these, the frequencies of the various partials remain constant relative to each other, again resulting in cheesy organ timbres. In addition, experience shows that a single LFO modulating all the ▶

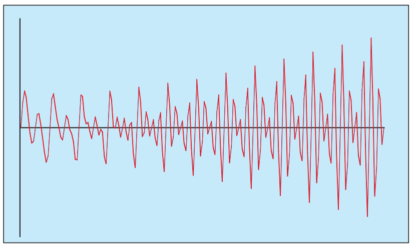


Figure 16: Applying the contours in Figure 15 to the sine wave oscillators in our additive synthesizer.

harmonics simultaneously will reinforce this cheesiness, so our analogue additive synth now needs to grow to gargantuan proportions with each oscillator boasting a pitch LFO, pitch envelope, amplitude LFO, and amplitude envelope.

Furthermore, and before we become completely carried away with simply generating timbres, we should also remember that music isn't just about creating sounds, it's about playing them, preferably with some sort of expression and character. So, to all of the above, we need to add some form of control for velocity- and pressure-sensitivity, and maybe some other real-time controllers. Now you have an analogue additive synth of which you can be proud. Sure, it's going to be nigh on impossible to crowbar the thing into your bedroom studio, but you hadn't intended to sleep there anyway. Had you?

## Now Let's Get Noisy

At this point, you may feel that your house-sized additive synth is complete. Unfortunately, it isn't, and I regret to inform you that — despite everything that I've written numerous times before — there are many sounds that you cannot break down into the sum of their sinusoids. At this point, I should reassure you that nothing we have discussed so far is actually wrong, it's just incomplete.

Consider the sounds of orchestral instruments such as flutes and trumpets. If you have the appropriate (expensive) equipment, you can separate their sounds into their component harmonics. However, if you then subtract these harmonics from the original sound there is a residual element: noise. This noise may not be very loud or intrusive, but it's there nonetheless. Consequently, many of your synthesized sounds will remain unconvincing if they lack a little noise within them. So our additive synth needs yet another sound source — a noise generator. Mind you, the noise produced by orchestral instruments is far from 'white' or 'pink'; it is heavily filtered by the nature of the instrument itself. So, despite everything, we need at least one filter in our additive synth. And this, of course, will need its own contour generator to ensure that the noise colour changes realistically over time. The noise generator will also need a VCA and its associated contour generator.

If this analysis seems a little arcane, it isn't. In fact, this extension to pure additive synthesis even has a name: if the analysis is performed beforehand it's called Spectral Modelling Synthesis. Without the signal analysis, you could just call it the 'sinusoids plus noise' model of sound generation (Figure 17).

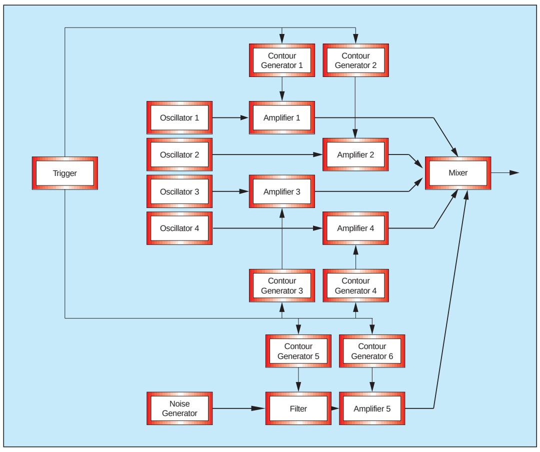


Figure 17: A 'sinusoids plus noise' synthesizer.

> **"While there's nothing stopping you from creating additive sounds using just a handful of oscillators, you're going to need dozens or even hundreds if you want to recreate natural sounds that contain a lot of overtones."**

## Don't Despair

Despite this potential complexity, simple additive synthesis is possible on quite modest analogue synths. So is 'sinusoids plus noise' synthesis. Indeed, I guarantee that anybody playing an instrument with two or more independently tuneable oscillators (and, maybe, a noise source) has created sounds employing tuned fifths, octaves, or whatever. As soon as you have done this, you've entering the weird and wonderful world of additive synthesis. So go and find a handful of extra oscillators, and get serious. Additives can be good for you, and it's great fun, I promise. SOS

# synth secrets

## PART 15: AN INTRODUCTION TO ESPS AND VOCODERS

**Gordon Reid** turns his attention to the effects that can be achieved when subtractive synthesis components are applied not to the output from oscillators, but to real-world sounds — such as human speech.

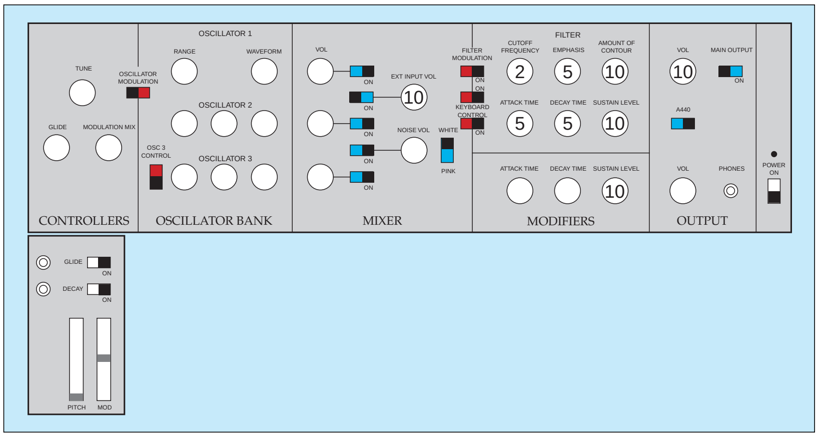


Figure 1: A typical setup for processing an external signal using a Minimoog.

Over the past few months I've described oscillators, filters, amplifiers and envelope generators — the devices that make up the architecture of a typical analogue 'subtractive' synthesizer. I've even demonstrated that you can use these to recreate methods of synthesis that are normally associated only with digital synthesizers. However, no matter how basic or how radical the concepts we've discussed, they have all shared one trait: the initial sound has been generated within the synthesizer. Usually, the synthesizer's internal oscillators have been the culprits, although self-oscillating filters have also reared their heads. Today, however, we're going to step beyond this limitation, and see how you can use external signals in subtractive synthesis.

### Getting The Signal In

Many of you will own contemporary analogue (and 'virtual analogue') synthesizers that offer signal inputs, but if you look at the back panels of vintage instruments you'll find that the idea is hardly a new one. If we return to the first non-modular synthesizers, we find that even the Minimoog offered a signal input as one of the five sound sources in its Mixer. Consequently, if you switched off the Moog's three internal oscillators and its noise generator, you could pass just the external signal through its filter and amplifier (see Figure 1).

It may look very simple in principle, but even this arrangement has numerous subtleties and pitfalls. For example, the Minimoog has no Initial Level control for its amplifier. This means no sound can pass through the synth until you press a key. However, when you press a key, the amplifier's gain increases according to the contour generator's settings, allowing you to chop the external audio into 'enveloped' bits and pieces.

In contrast, if the filter's cutoff frequency control is greater than zero, some low frequencies will pass through it regardless of whether you press a key or not — though you still won't hear anything until you press a key, because the amplifier's gain is zero. When you press a key, the filter's dedicated contour generator will alter the cutoff frequency (or not, according to the Amount control) thus allowing a greater or lesser amount of the external signal's spectrum through the Moog. Furthermore, you can use the filter's resonance to emphasise parts of the external signal's spectrum, just as you would if the internal oscillators generated the sound. The advantage of this is obvious: whatever the nature of the external signal, you can use the Minimoog's filter to create new timbres.

Despite these abilities, the Minimoog is still very limited in the way that it can process an external signal. After all, there's no way to determine the pitch of the sound passing through it, nor can you determine when you hear the sound, other than by pressing a key. What we want is the ability to control the synthesizer by applying an external signal — maybe using a guitar to 'play' it, or using your voice to make it 'speak'. So let's introduce the two 'modules' that make this possible: the envelope follower and, first, the pitch-to-voltage converter.

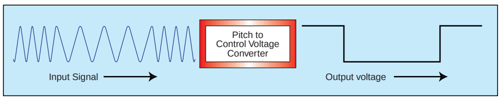


Figure 2: A simple form of pitch/CV converter.

### External Signal Processing (1): The Pitch-to-Voltage Converter

Let's look at the idea of controlling and playing your synthesizer using a guitar. Oh yes... and don't forget, this is pure analogue technology we are considering, so we can't cheat by introducing a MIDI-to-CV converter! What we need instead is the

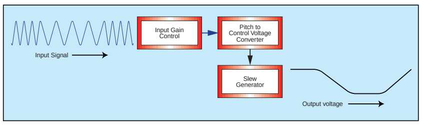


Figure 3: A more sophisticated pitch/CV converter.

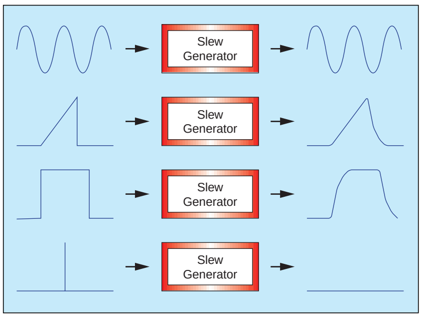


Figure 4: Examples of a slew generator in action.

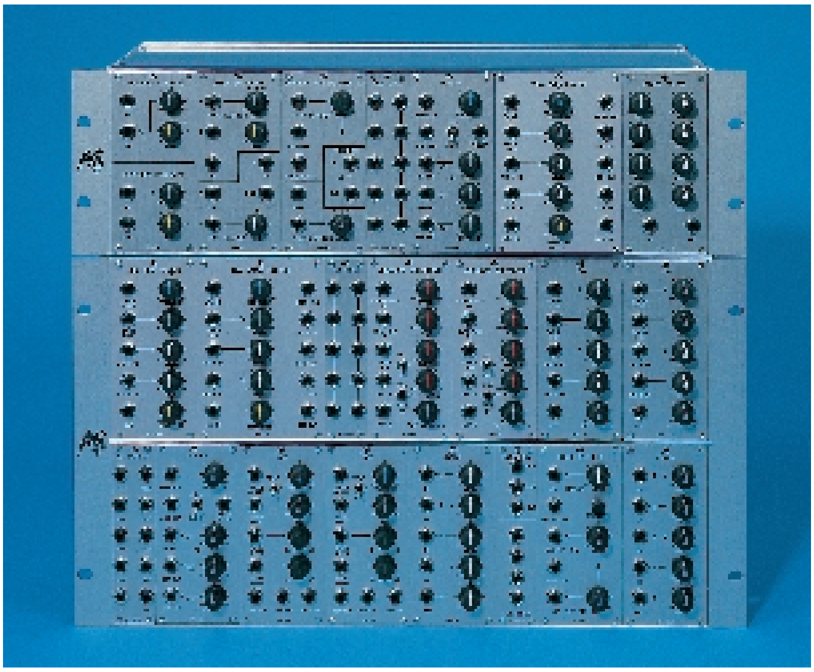
aforementioned pitch-to-voltage converter. More commonly called a pitch/CV converter, this is a device that accepts a monophonic signal (ie. a signal with only one pitch present) at its input, determines its pitch, and then produces an appropriate control voltage at its output (see Figure 2).

Neat, isn't it? Indeed, if everything were as elementary as this, the worlds of electronics and synthesis would be very simple indeed. Unfortunately, pitch/CV converters can be fooled by stray signals and background noise, causing glitching. To overcome this, we add two sub-modules. The first of these is a simple audio amplifier called an input level or input gain control. This ensures that the external signal enters the converter itself at an optimal level. The second is a Slew Generator, or 'Lag Processor' (see Figure 3).

We haven't encountered slew generators before in Synth Secrets, but their effect might be obvious from the figure. In essence, they slow down transitions from one voltage to another, thus 'slewing' any abrupt changes over a period. (In a sophisticated system, the amount of slewing will be governed by a Slew Rate control.) As you can see from Figure 4, slowly changing signals (such as the low-frequency sine wave) pass unaffected, whereas signals with sharp transitions become rounded. Interestingly, the single, sharp spike disappears completely. OK, this is an idealised description, but it gives you the general idea.

You should now be able to see that the slew generator is simply a low-pass filter, albeit one with a handful of specialised uses. (On most analogue synthesizers, it's a 6dB/oct low-pass filter with cutoff frequency variable in the range 0Hz to approximately 1kHz.) ▶

Modular synthesizers, such as the Analogue Systems Integrator below, allow you to configure as many signal processors as you wish (or can afford). There's nothing stopping you building a complete vocoder if you have sufficient envelope followers, filters and VCAs (plus a few other bits and pieces such as oscillators, mixers, and CV offset controllers).



You will most commonly encounter a slew generator in its role as a 'portamento' circuit, smoothing the transition from one keyboard-generated pitch CV to another. However, when it's hooked up to a pitch/CV converter, the slew generator's *raison d'être* is to remove the inevitable glitches that occur when the pitch detector loses lock on the desired signal. (Without the slew generator, the output CV would jump around wildly until lock was re-established.)

Moving on, we can enhance Figure 3 and improve the converter's performance still further. To be specific, we add a band-pass filter to create a narrow 'pass band' of accepted frequencies (see Figure 5). This reduces the risk of extraneous signals or high-amplitude harmonics confusing the pitch detector.

You might think that we now have everything needed to control our synthesizer using a guitar, but even the sophisticated converter in Figure 5 is not sufficient for this. Yes, it provides a monophonic CV that you can use to determine the oscillators' pitches, but it tells the synth nothing about the changing loudness of the notes (or even when they occur). For this we need something completely different...

## External Signal Processing (2): The Envelope Follower

We have already determined that the pitch/CV circuit will provide a CV for the oscillators, so we now need something that can provide a CV to control the synthesizer's VCF and/or VCA. This something is an Envelope Follower (strictly speaking, a 'peak amplitude follower') — a circuit that measures the amplitude of the positive peaks of the waveform.

If you're interested in the electronics (and because it's such an elegant solution) take a peek at the simple envelope follower in Figure 6. This works in a deliciously simple fashion. If successive peaks of the input signal are of increasing amplitude, the capacitor is charged up, and if the peaks are decreasing in amplitude the capacitor discharges at a rate determined by the value of the variable resistor. Of course, there is a small discharge between successive peaks even if the amplitude of the overall envelope is increasing but, if you choose your component values carefully, you can create the output shown in blue in Figure 7. As you can see, this is remarkably similar to the true signal 'envelope'.

As with our pitch/CV converter, there's nothing stopping us from making the envelope follower more sophisticated, again adding an Input Gain Control and a slew generator to improve its performance. The latter of these will smooth out the 'bumps', making the output CV even more like the envelope of the signal (Figure 8).

## Putting It All Together

Now let's put it all together. Figure 9 shows an external signal — which can be the output from a guitar, a microphone, a CD player, or whatever — split and directed down two signal paths. The four blocks in the upper path are those shown in Figure 5, whereas the lower path is provided by Figure 8. Note that in the following figures I have used blue arrows for audio signals and black arrows for control voltages. I hope that this makes things clearer.

Now look at the CVs' destinations. Clearly, the pitch of the input is controlling the pitch of the oscillator, whereas the loudness of the input is controlling the cutoff frequency of a VCF and the gain of a VCA. The envelope follower is, therefore, replacing the contour generators that you would find in a conventional configuration. So, while the oscillator is providing the basic timbre of the output, the incoming signal is determining the pitch as well as articulating the new sound.

So there we have it... the perfect way (in theory) to control your synthesizer using an external signal such as a guitar. But while Figure 9 offers some interesting musical possibilities it is limited in one important way: the external signal has a very limited ability to modify the timbre of the output. In fact, the only way to affect the timbre is by patching the pitch CV or loudness CV to the cutoff frequency input of the filter. So, is there a way to make your external signal determine both the loudness and the timbre of the synthesized sound? Of course there is, and you probably have an example tucked away in one of your synthesizers or effects units. It's a vocoder.

## Introducing Vocoders

Imagine playing a percussion track — from tape, or from a rhythm machine — into an envelope follower. As you might guess, the follower will generate a succession of decaying pulses that you can use as envelopes or as triggers. If you then play another external signal such as a sustained organ chord through a VCA that is controlled by a contour generator that is itself triggered by the envelope follower... well, I'm sure that you get the idea. The organ now 'plays' the chords rhythmically and in perfect synchronisation with the percussion track, whether you lift your hand from the keys or not! (See Figure 10.)

Extending this idea further, you

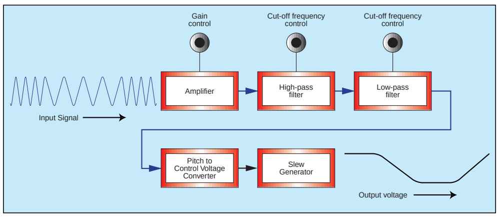


Figure 5: A sophisticated analogue pitch-to-CV converter.

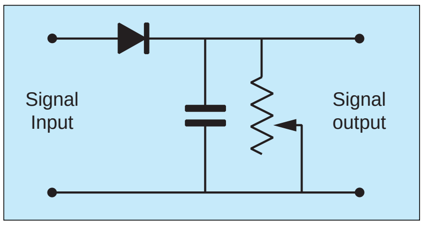


Figure 6: A simple analogue envelope follower.

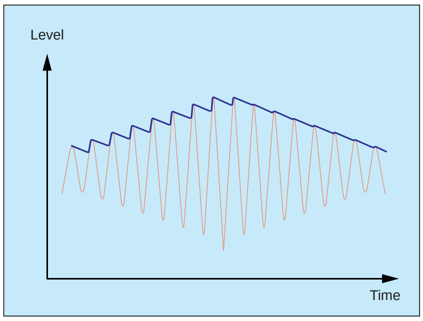


Figure 7: The output from an envelope follower.

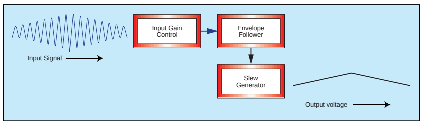


Figure 8: A more sophisticated envelope follower.

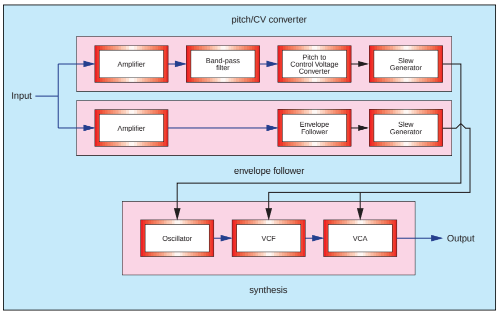


Figure 9: An external signal controlling a simple analogue synthesizer.

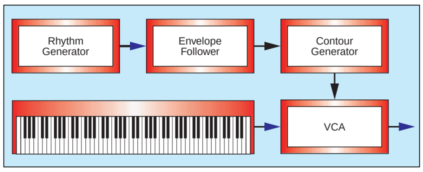


Figure 10: The cheat's way to perfect keyboard rhythm.

could attach a microphone and use your voice to articulate your keyboard playing — allowing sound through the VCA when you say (or sing) words, but creating silence between words (Figure 11). Unfortunately, the envelope follower is simply following the amplitude peaks of what you say or sing... the words themselves are completely irrelevant. To put it another way, it doesn't matter what frequencies are present in the signal, it is only the total amplitude that determines the output.

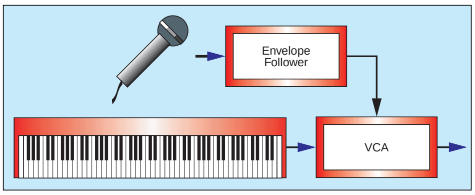


Figure 11: Articulating a keyboard sound using your voice.

To understand how great a limitation this can be, imagine that you have a signal with no frequencies present below 1kHz but so much signal above 1kHz that it registers as the maximum that the circuit can handle. In this scenario, the output from the envelope follower will also be a maximum. Now let's turn this example on its head: there is no signal above 1kHz, and maximum below this frequency. The envelope follower will again produce a maximum CV output. Ouch! Despite the fact that the signal in the first example contains only high frequencies while the second contains only low frequencies, the result is the same.

So, here's an idea: let's split the signal into two paths, and place two filters before a pair of envelope followers (see Figure 12). In this scenario, low-frequency signals cause Envelope Follower 1 to generate a CV, while high-frequency signals cause Envelope Follower 2 to generate a CV. If we now send these CVs to a pair of VCAs, we can configure our synthesizer so that its response to incoming signals is 'frequency-sensitive'. For example, we can use the frequency content of the input signal to determine the relative amplitudes of the signals generated by two oscillators (Figure 13). Intriguing, yes?

## The Genesis Of The Vocoder

**A research physicist named Homer Dudley invented the Vocoder (the VOice EnCODER or, according to some commentators, the Voice Operated reCOrDER) when he was working at Bell Laboratories, New Jersey, in 1939. He developed the machine as a research device for experimenting with audio compression, primarily to improve the voice-carrying capabilities of his employer's telephone lines.**

**Following WWII, Dudley visited Bonn University and met a chap named Werner Meyer-Eppler who, at the time, was the Director of Phonetics at the University. Meyer-Eppler recognised the relevance of the vocoder to electronic music, and subsequently used it within a number of compositions that would eventually become the inspiration for the German 'Electronische Musik' movement. Understandably, the fidelity of Dudley's vocoder was limited, but the fundamental concept has remained unchanged to this day.**

However, we don't necessarily need to use multiple VCOs to generate interesting effects. Instead, we could take a complex signal and pass this through a second bank of band-pass filters to split groups of harmonics into a number of separate signals, each occupying a defined band of frequencies.

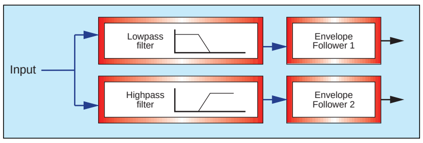


Figure 12: A frequency-sensitive envelope follower.

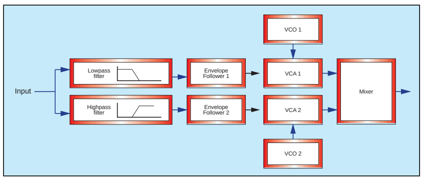


Figure 13: Using frequency to select between two oscillators.

Now look at Figure 14. As before, the envelope followers raise and lower the gains of the appropriate VCAs in sympathy with the frequency content of the input (this, by the way, is a 'Spectrum Analyser'). However, instead of controlling the amplitudes of signals generated by independent oscillators, the VCAs now control the amplitudes of the harmonics in each of the frequency bands.

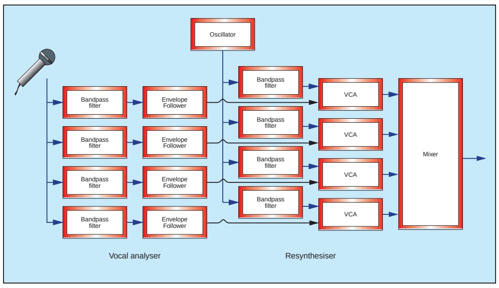


Figure 14: A simple 4-band vocoder.

This is a hugely important result. Imagine that the signal presented to the envelope followers is your voice (we will call this the modulator) and that an oscillator generates the signal presented to the second bank of band-pass filters (we will call this the carrier). In this case, the carrier provides the basic tonality of the output, but the modulator determines its frequency content and amplitude. In other words, the modulator articulates the carrier. Voila! We have designed a vocoder.

## More Advanced Vocoding

Many vocoders generate an internal carrier wave — usually a sawtooth because it's the most harmonically rich of the simple waveforms, although a rounded pulse wave would be more suitable for vocal sounds, because this is closest to the raw waveform produced by the human vocal chords. An internal white noise generator is another good choice for speech resynthesis, because this contains all the frequencies in the sound spectrum. However, better vocoders offer two external signal inputs — the first for the modulator, and the second for the carrier (see Figure 15).

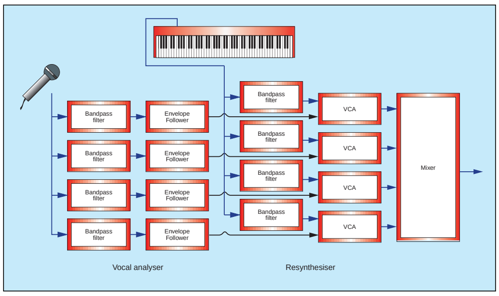


Figure 15: Using external signals for both carrier and modulator.

Of course, there's nothing to say that the modulator must be a vocal signal. You can use guitars, other keyboards, any acoustic instrument (provided, of course, that you convert its sound into an electrical signal using a microphone or transducer) or even the outputs from CD players and the radio. Likewise, the carrier can be any signal. Indeed, using the same signal as both

A darling of the dance and industrial scene, the Korg MS20 has a powerful External Signal Processor that offers both a pitch/CV converter and an envelope follower.

carrier and modulator provides one of the most interesting vocoder effects. If you want an even more radical configuration, you could even place pitch/CV converters and oscillators at one or both of the inputs.

Before finishing, I would like to describe two final enhancements to our vocoder. If you look at the front panels of units such as the Roland SVC350, you will see a bunch of faders. These scale the CVs produced by the envelope followers, allowing you to tailor the vocoder's response, accentuating or attenuating the outputs of specific bands. Less visibly, but no less importantly, some vocoders (the Roland VP330 springs to mind) replace one of the band-passed carrier signals with a noise generator. This is very important for correct articulation of sibilants and consonants — the short, noisy sounds (for example, the letters 'd', 't', and 's' without their attendant vowel sounds) created primarily by your lips and tongue rather than the vocal chords. We can add the front panel controls and noise generator as shown in Figure 16.

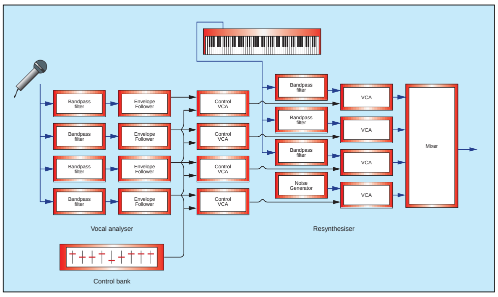


Figure 16: Modifying the vocoder's response.

OK, even with these enhancements, there are limitations. In particular, the low resolution of the band-pass filters — typical roll-offs are 6dB/oct or 12dB/oct — ensures that the output articulation retains only a remote semblance of the original. But with sufficient bands (10 plus noise is enough) a vocoder is easily good enough for 'Mr Blue Sky'!

So that's it. Easy, huh? Well... yes it is. When Walter Carlos recorded the score for *A Clockwork Orange*, he used off-the-shelf Moog filters, oscillators, envelope followers and VCAs to resynthesize (ie. vocode) the choral sounds. Given the cash, space and patience, you could do the same. So here's this month's Synth Secret...

*If your synthesizer has an external signal input plus an envelope follower and a pitch/CV converter, it is a much more powerful and flexible instrument than it would otherwise be. You should experiment with them!*

Alternatively, you could simply stick with conventional emulations of woodwind, strings and brass, or continue to create silly bleeping and squelchy noises. But where's the fun in that? SOS

# synth secrets

## PART 16: FROM SAMPLE AND HOLD TO SAMPLE-RATE CONVERTERS (1)

**Gordon Reid** introduces the synthesis modules that allow you to create a number of commonly used 'random' effects, and their close relatives — analogue sequencers.

You might think that after more than a year's worth of articles about oscillators, filters, amplifiers, LFOs, contour generators, external signal processors, and heaven knows what else, we would have exhausted the modules that make up an analogue synthesizer. Not a bit of it! This month, we're going to start with a common synth component, the Sample and Hold module, and consider its use with a few less common ones. This potentially obscure discussion will nevertheless take us from Emerson, Lake and Palmer to Donna Summer, and from clock pulses to the the threshold of analogue-to-digital conversion. Not bad, eh?

### Not All Samples Are Digital

Given that it's not necessary for generating any conventional imitative sounds, the Sample and Hold (S&H) module is remarkably common. Early Moog modulars had them, as did classic synths such as the ARP 2600 and Odyssey, so many of the characteristic sounds they make possible have entered common usage. To understand S&H, take a peek at Figure 1 (right). This is another of my 'remarkably simple' circuits, comprising just two components: a capacitor, and a switch.

Imagine, if you will, that a standard synthesizer signal is presented to the input on the left of the diagram. If doesn't matter whether this is an audio signal, an LFO, an envelope, or anything else. Now imagine that, just for an instant, the switch closes. Provided that the capacitor can react quickly enough, it then charges up (or discharges down) to the voltage at the input, thus 'sampling' that voltage. Then, once the switch has opened again, the voltage across the capacitor cannot change. This is because on the left-hand side there is no circuit and, on the right-hand side, the impedance is infinite (which means that no current can flow). However, although no current flows, you can still measure the voltage across the output. (Of course, the impedance is never truly infinite, so the voltage will decay slowly, and one measure of the quality of an S&H module is the slowness of the rate at which this decay occurs.)

So that's all there is to it... when the switch is closed, the capacitor 'samples' the input voltage. When the switch is open, the capacitor 'holds' that voltage, allowing other circuits to respond to it as appropriate. Sample and Hold... simple, yes?

### Clock Generators

Now, if you were limited to closing the switch in Figure 1 manually, the S&H would not be of much use. So synthesizers have electronic switches, and provide another module that is capable of opening and closing them at high speeds. The most common such module is one that we have not yet discussed in Synth Secrets. It is the Clock Generator.

As you might guess from its name, a Clock Generator provides an evenly spaced stream of 'ticks' in the form of short pulses (see Figure 2 below). Of course, this is simply a specialised form of oscillator: one that produces a pulse wave at, typically, subsonic and low audio frequencies. Most often, you would use the clock as a timer, for example, triggering envelopes or as a 'sync' or reset source for conventional audio oscillators and LFOs. However, those are topics for another day.

Today's use for the Clock Generator is to provide a stream of very short pulses that trigger the switch in Figure 1. In other words, when the pulse is On, the S&H circuit samples, and when the pulse is Off, the S&H holds. As you can see, I have shown the clock ticks as +10V pulses with negligible width. In practice, many S&H circuits will perceive a trigger when they receive any positive-going waveform with a sharp leading edge.

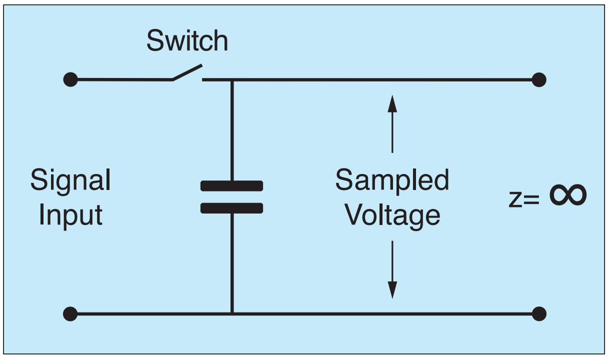


Figure 1: The simplest representation of a S&H circuit.

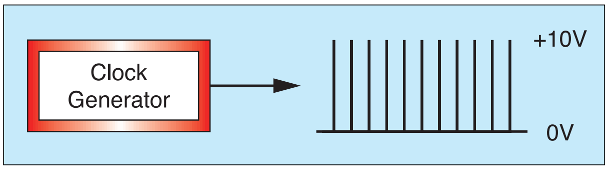


Figure 2: The output from a Clock Generator.

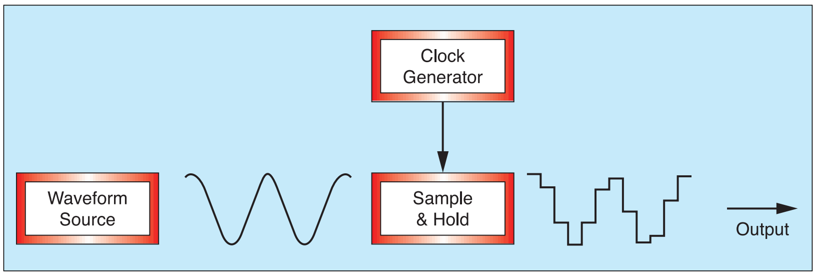


Figure 3: A simple example of S&H.

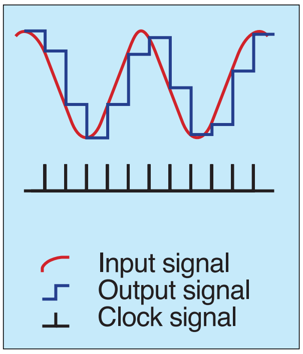


Figure 4: Explaining S&H.

If we marry our clock generator to the S&H circuit, we can devise a composite module that incorporates two sub-modules: the clock and the S&H circuit itself (see Figure 3 above).

Let's analyse what's happening in Figure 3. It shows a sine wave entering the signal input of the S&H module. At the same time, a clock provides a stream of pulses that it presents to the S&H's trigger input. The output produced by the S&H circuit is then the 'rectangular' waveform shown in the Figure.

If the reason for this output is not clear, Figure 4 should clarify things. Each time the S&H module receives a trigger from the clock generator, it measures (or 'samples') the voltage of the input signal (shown in red). It then holds this voltage (the blue line) until it receives the next trigger, at which point it repeats the operation. It 'samples' and it 'holds', just as we've discussed.

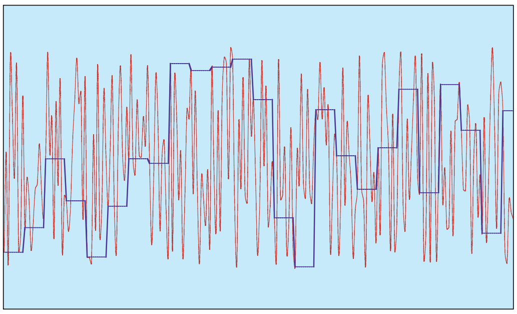


Figure 5: A waveform produced by passing a random (noise) signal though an S&H module.

This result would not be very interesting if a sine wave was the only signal you could present to the S&H's audio input. Fortunately, the input signal can be anything: a synthesized audio waveform, a slowly varying CV, or even an external signal such as the sound from a CD player or of an instrument being played. However, most synthesizers also have a noise generator that produces a 'random' signal, and this is the most commonly used type of S&H input. Why? Because it creates a number of musically pleasing effects.

Figure 5 (below left) shows what happens when you S&H a noise signal. Again, the input is the red line, with the output shown in blue. Now, what can we do with this output? The first thing to remember is that there's nothing stopping us from directing the output signal to the inputs of any other synthesizer modules, whereupon it will modify any other CVs or signals within the synthesizer. However, there are two common uses that you will have heard many times, and which deserve special mention here.

The first involves directing the S&H output to an oscillator's CV input so that it makes the pitch vary randomly. Interesting on an analogue synth, this facility comes into its own on a digital synth. This is because — with a suitable amplifier to attenuate the variations — you can use this effect to add delicate random fluctuations to the pitch of a sound. Far less obvious (and therefore more interesting) than cyclic LFO modulation, it's the key to humanising many digital sounds, and it's absolutely vital to any impersonations of real instruments that waver slightly as the note plays. For this reason, many LFOs offer a 'random' output waveform alongside the sine, pulse and triangle waves that most provide.

The second example requires you to direct the S&H output to the CV input of a low-pass filter. This affects the high-frequency content of the sound and, therefore, each clock pulse changes the timbre of the audio passing through the filter. If, like me, you grew up with 'First Impression' from ELP's *Karn Evil 9* ringing in your ears, you'll know this sound intimately. Indeed, this combination of clock, S&H and noise (see Figure 6, left) is so deeply routed in synthesis that some synthesizers combine them in a single module. The RS Integrator module in Figure 7 (above) demonstrates this. At the top of its panel, you'll find a noise generator with an associated level control and output. Beneath this, you have the signal input for the S&H, the clock input, the S&H output, and a level control that adjusts the gain of the output signal. Below this, you have the clock itself, with a frequency control and output. There is just one control that is less than obvious: the 'one shot' button allows you to trigger the S&H switch manually.

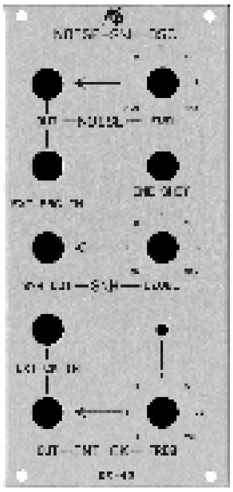


Figure 7: A combined Noise, S&H and Clock module.

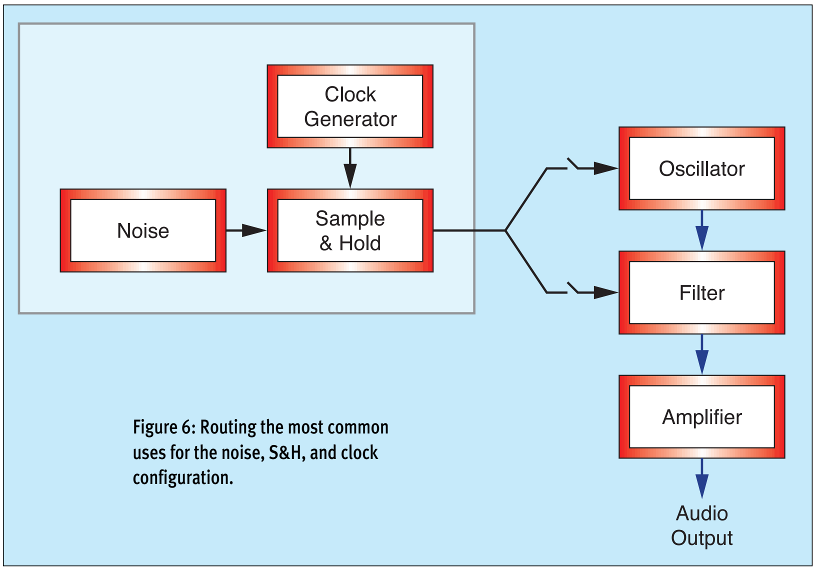


Figure 6: Routing the most common uses for the noise, S&H, and clock configuration.

Unlike some other synthesizers with prepatched S&H modules, the three sub-modules in the RS40 are not pre-connected 'behind the scenes' (if they were, the 'one shot' button would not work). This means that you must patch sources into both the EXT SRC IN and EXT CLK IN sockets. The simplest and most obvious sources for these purposes are the noise generator and clock outputs that lie immediately above and below them but, as suggested above, you are by no means limited to these.

Before moving on, we should look at two further refinements to the S&H concept. The first involves the nature of the signals that will trigger the sampling operation. If you think about the pulses produced by the clock generator you will realise that they can not be truly instantaneous — they have a duration. So, is the sample taken on the leading edge of the pulse, on the trailing edge, or (less usually) on both edges? On analogue synthesizers, it's usually the leading edge that does the trick. Nevertheless, if you want to experiment with trailing-edge triggering you can do so if you invert the trigger signal and provide a suitable DC offset.

The second refinement is even more esoteric. Let's face it, you won't often trip across a Buchla Model 264 Quad Sample and Hold/Polyphonic Adaptor (on which this feature appears) at your local car-boot sale. However, you'll find the same features on one of the most common patchable analogue synthesizers of them all. The effect is 'Track and Hold' and the common synth is the Korg MS20.

Of course, Buchla and Korg didn't silkscreen the name 'T&H' on their modules; they called them S&H modules like everybody else. However, there's a difference, and it lies in way that the circuit handles the clock pulse. Unlike everything discussed above, sampling is not triggered by an edge of the clock pulse and then held until the next equivalent edge. Instead, the circuit is transparent when the pulse is high, and the hold is initiated by the falling edge, remaining constant until the clock goes high again. As you might imagine, this means that you can use the duty cycle of your clock pulse to determine the proportion of time that the output (in blue) tracks the input (in red), and the proportion that it holds (see Figure 8).

T&H modules are powerful tools that provide many possibilities not achievable using conventional S&H modules. For example, you can use one together with one of last month's Envelope Followers so that the amplitude of an external audio signal determines whether the module Tracks or Holds. Interesting, yes? Furthermore, you can use the T&H as a conventional S&H if you use a clock with a pulse of minimal duration.

## Introducing Sequencers

There are many other enhancements to the basic S&H concept, and other modules — such as pulse shapers — that you can add to generate specific effects. Nevertheless, while an S&H module is very useful for creating random effects, and for adding that ineffable sparkle to many sounds, it is not easy to create melodies using one. Nor will an S&H easily provide conventional movement within your sounds. For this, we need a module that allows you to *decide* the value of its output CV at every clock step. We call such a device an analogue sequencer.

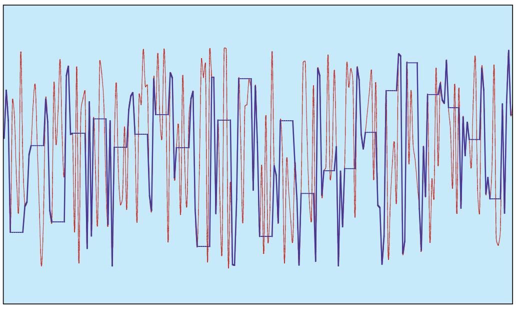


Figure 8: A Track & Hold input/output graph.

Figure 9 (below) shows the simplest form of analogue sequencer (which, henceforth, I'll simply call a 'sequencer'). It has 16 steps, and a dedicated potentiometer, controlled by a fader, determines the output at each step. A clock generator (which may be internal or provided by an outside source) causes the sequencer to step from one fader to the next, and from step 16 back to step 1 when it reaches the end of the row. Of course, the potentiometers could be knobs — and they often are — but it's much easier to see what's happening using faders.

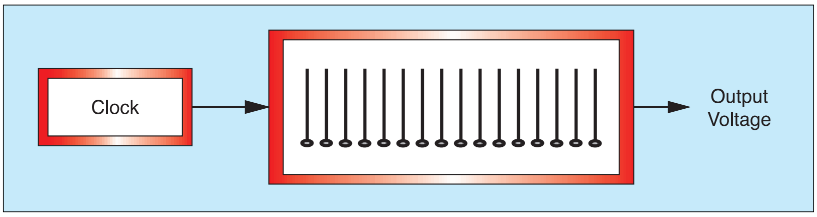


Figure 9: A simple analogue sequencer.

Let's suppose that the output is 0V when a fader is at the bottom of its travel, and +2V when it is at the top. If you then direct the output voltage to the pitch CV input of a 1V/octave oscillator, you can determine the pitch anywhere in a two-octave range. This makes it possible to construct simple 16-note melodies that repeat continuously while the clock is running.

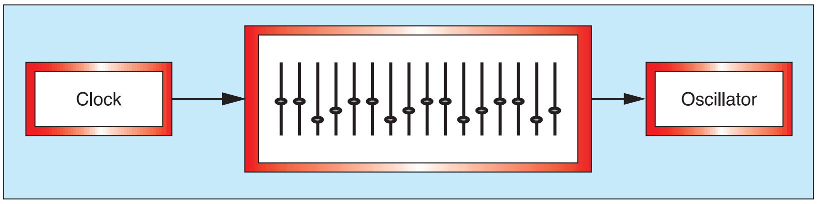


Figure 10: A typical '70s disco bass sequence.

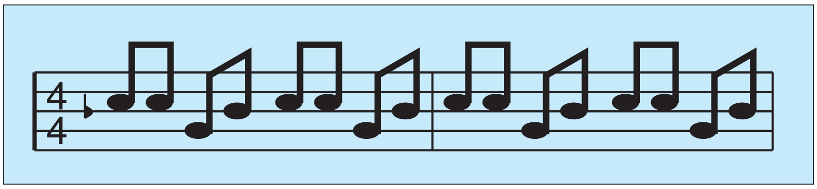


Figure 11: The same sequence written in traditional notation.

Look at Figure 10 on page 200. If you've used a sequencer extensively, you'll recognise it immediately. It's the 'do-do de-de da-da de-de' bass line from thousands of best-forgotten disco hits inspired by Donna Summer's 'I Feel Love'. If you look at Figure 11 (also on page 200), you'll see that the configuration of the faders on the sequencer exactly mimics the notes as they would appear on a musical staff. The similarity is not surprising, given that the two systems use identical concepts for representing musical pitch.

In reality, sequencers are often far more complex than the one shown in Figure 9. A powerful unit will offer multiple rows of CV generators, multiple outputs, multiple clock inputs for the rows, CVs for the clocks themselves, range controls for the CVs, note-skip capabilities, variable sequence lengths, and many other facilities (see Figure 12, above for an example). However, the underlying concept remains the same.

## Voltage Quantisation

Despite the power of the sequencer shown above, it's a very laborious task to make it create musical sequences. Why? It's because the faders (or, in this case, the knobs) are capable of producing any voltage within their range — including values that lie between the notes in a conventional musical scale. This means that, instead of the 25 semitones that exist in the full two octaves discussed above, or the 121 that lie within the 0V to +10V range of the sequencer in Figure 12, there are thousands, maybe millions, of discrete voltages available. This makes precise tuning almost impossible.

The solution to this problem lies in a module called a Voltage Quantiser, or simply a 'quantiser' — a device that rounds every voltage passing through it up or down to that which produces an exact semitone. It does this by making sure that, no matter what the input voltage may be, the output is one of the voltages that define the well-tempered scale for a 1V/Octave synthesizer (see Table 1 and Figure 13, right). As you can see, the 'spacing' between the voltages of the notes is 1/12th of a volt (although the relationship between the frequencies is not so simple).

The presence of the quantiser makes it much easier to set the controls on the sequencer to create recognisable melodies or bass lines. Similarly, you can use it to quantise the output from your S&H module to be in semitones (ie. 'real' notes) rather than random pitches. You can even use it to convert a continuously rising or falling CV into a 'stepped' one — an effect that we often use to convert portamento into glissando (see Figure 14, opposite). As you can see, this is very similar to the output of the S&H module in Figure 3. The difference is that, whereas the S&H measures a new voltage every time it receives a trigger from an outside source, the quantiser

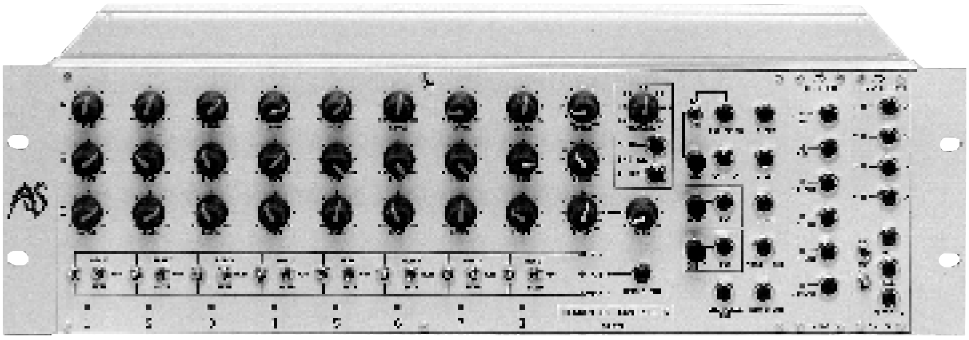


Figure 12: A powerful analogue sequencer.

| Note Name | Frequency (Hz) | Minimum Input Voltage | Maximum Input Voltage | Output Voltage |
|---|---|---|---|---|
| C2 | 131 | 0.958 | 1.042 | 1.000 |
| C#2 (Db2) | 139 | 1.042 | 1.125 | 1.083 |
| D2 | 147 | 1.125 | 1.208 | 1.167 |
| D#2 (Eb2) | 156 | 1.208 | 1.292 | 1.250 |
| E2 | 165 | 1.292 | 1.375 | 1.333 |
| F2 | 175 | 1.375 | 1.458 | 1.417 |
| F#2 (Gb2) | 185 | 1.458 | 1.542 | 1.500 |
| G2 | 196 | 1.542 | 1.625 | 1.583 |
| G#2 (Ab2) | 208 | 1.625 | 1.708 | 1.667 |
| A2 | 220 | 1.708 | 1.792 | 1.750 |
| A#2 (Bb2) | 233 | 1.792 | 1.875 | 1.833 |
| B2 | 247 | 1.875 | 1.958 | 1.917 |
| C3 | 262 | 1.958 | 2.042 | 2.000 |
| C#3 (Db3) | 277 | 2.042 | 2.125 | 2.083 |
| D3 | 294 | 2.125 | 2.208 | 2.167 |
| D#3 (Eb3) | 311 | 2.208 | 2.292 | 2.250 |
| E2 | 330 | 2.292 | 2.375 | 2.333 |
| F3 | 349 | 2.375 | 2.458 | 2.417 |
| F#3 (Gb3) | 370 | 2.458 | 2.542 | 2.500 |
| G3 | 392 | 2.542 | 2.625 | 2.583 |
| G#3 (Ab3) | 415 | 2.625 | 2.708 | 2.667 |
| A3 | 440 | 2.708 | 2.792 | 2.750 |
| A#3 (Bb3) | 466 | 2.792 | 2.875 | 2.833 |
| B3 | 494 | 2.875 | 2.958 | 2.917 |
| C4 | 523 | 2.958 | 3.042 | 3.000 |

Table 1: Part of the range of logarithmic frequencies and linear voltages that represent the notes between C2 and C4.

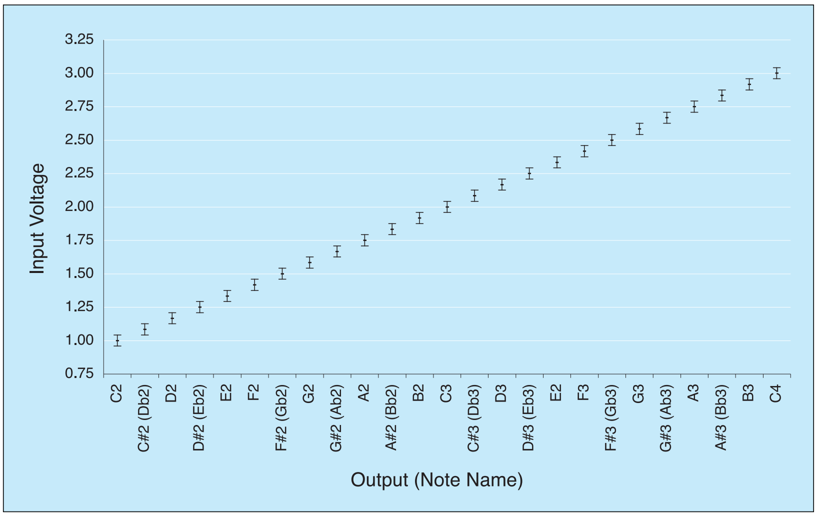


Figure 13: Any voltage between the maximum and minimum is corrected to the exact note value.

steps between voltages every time the input moves between ranges of values.

## Programmable Scale Generators

There's an enhanced type of quantiser that's also worthy of special mention. This is the 'Scale Generator' (or Scale Quantiser), a module that quantises the voltage in such a way that the output conforms to a predetermined musical scale. A programmable scale generator takes this idea one stage further: while it may include predetermined scales as options, it also allows you to choose the notes that define any desired scale. This leads to an unexpected combination of modules, and one that produces interesting musical effects. Figure 15 shows how it works.

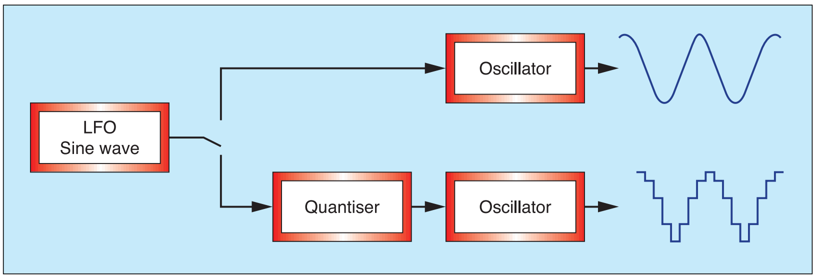
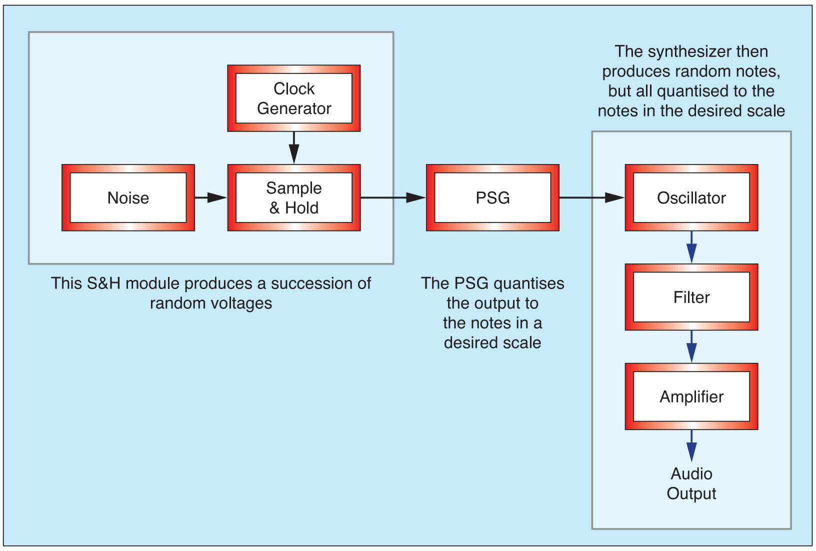


Figure 14: Converting a smoothly varying voltage into a stepped one.



This S&H module produces a succession of random voltages

The PSG quantises the output to the notes in a desired scale

The synthesizer then produces random notes, but all quantised to the notes in the desired scale

Figure 15: Producing a random series of notes in a desired scale.

Firstly, the S&H module produces a succession of random voltages. The programmable scale generator (the 'PSG') then quantises these voltages to conform to the notes in a desired scale. You direct the output from the PSG to the pitch CV input of an oscillator which, as a consequence, produces random notes — but always within that desired scale. This is particularly pleasing if you make the defined scale an arpeggio because you then have a random arpeggiator — one of my favourite effects, and still available only on a handful of synths.

## Slewing The Output

So far this month, we've only discussed the creation of stepped voltages from continuously varying signals. But there are many times when you would like the transitions from one voltage to another to be a little less abrupt. So let's finish by looking at ways in which we can smooth our S&H or sequenced output.

When you press a key on a voltage-controlled synth, you provide a voltage sample to an oscillator's pitch CV input. You specify the voltage itself by choosing which key to play. Then, when you press a second key, you present a second sample to the oscillator's input, and the pitch jumps almost instantaneously to that defined by the new voltage. But if you insert a simple Slew Generator (see last month's instalment of this series) into the keyboard CV signal path, you smooth the transitions at the oscillator's CV input, thus making the pitch glide from one note to the next. This, of course, is portamento.

Figure 16, below, shows the change that a simple slew generator will make to an S&H signal. As you can see, it rounds the transitions and creates a characteristic 'shark's tooth' pattern in the output waveform. We obtain this shape because the rate of transition between states is much lower that the slew generator's cut-off frequency, and because I've depicted the slew as an exponential glide between voltages, as it would be on most vintage synths. But what happens if we increase the speed of the S&H clock, or change the nature of the filter so that the signal is 'smoothed' in a more conventional way? Figure 17 shows what happens when we smooth a stepped voltage using a sophisticated filter that has a response more in keeping with the sample rate.

But look again: is the red line the input from an S&H module, with the blue line the smoothed output? Or is the blue line an audio signal input, with the red line the S&H output? Tricky! And it's the question that, in one leap, takes us all the way from the analogue domain to the digital domain and back again. It's a leap that we'll take next month. SOS

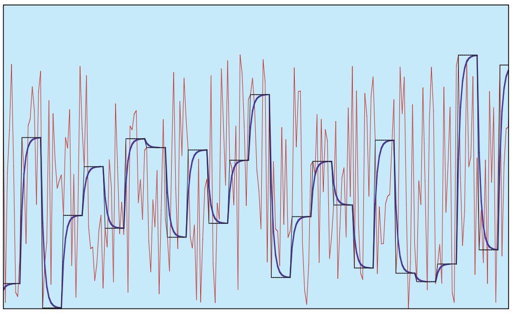


Figure 16: A slewed S&H output.

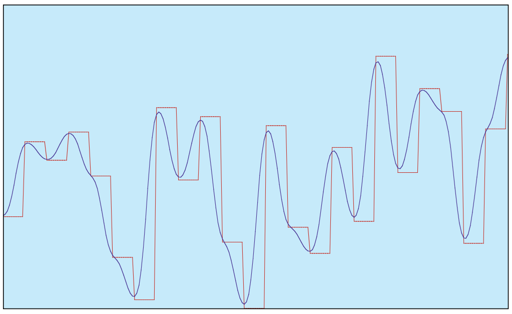


Figure 17: Filtering the S&H output using a powerful audio filter.

# synth secrets

## PART 17: FROM SAMPLE AND HOLD TO SAMPLE-RATE CONVERTERS (2)

Do you remember Figure 1? It featured in last month's Synth Secrets, and when I presented it to you, I asked the following question: Is the red line the input from an S&H module, with the blue line the smoothed output? Alternatively, is the blue line an audio signal input, with the red line the S&H output? This month we're going to find out the answer.

Sample and Hold modules, as **Gordon Reid** explained last month, convert a continuously varying signal into a stepped series of fixed pitches. And this, as we shall see, is the basis of what we know as 'digital audio'...

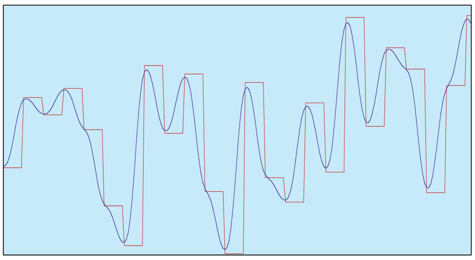


Figure 1: Smoothing a stepped waveform (or *vice versa*?).

### Generating Step Sequences

In Synth Secrets 16 we talked a lot about Sample and Hold modules and step sequencers, two devices that generate 'stepped' CV signals at their outputs. So let's start here, and consider the 13-step sequencer shown in Figure 2. OK, I admit that I've never heard of a 13-step sequencer, but that's not relevant — most 16-step devices allow you to truncate the sequence at any step you wish. Anyway, let's set up our sequencer as shown in Figure 3...

I have set the faders on the sequencer so that (hypothetically) it outputs 1V at step one, 1.083V at step two, 1.167V at step three, and so on up to precisely 2V at step 13. The eagle-eyed among you will notice that this quantised output represents a chromatic scale from C1 to C2 on a 1V/octave synthesizer. To ensure that the output conforms precisely to these voltages, I have inserted a quantiser between the sequencer and the oscillator it controls. We can then represent the output as shown in Figure 4.

Now, let's hypothesise that the clock runs at one pulse per second. Clearly, the oscillator will step between notes once per second, completing the scale once every 13 seconds before resetting to the beginning and starting again. But what happens if we accelerate the clock? At 130 pulses per second, the scale repeats 10 times per second, creating a complex tone at the output.

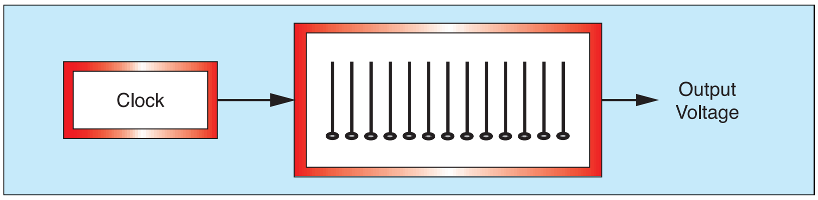


Figure 2: A sequencer with 13 steps.

If you try to accelerate a conventional synthesizer clock much further, you'll soon reach its limit. This is because clocks are low-frequency oscillators, most of which have a maximum frequency of just a few hundred Hertz. So let's replace the dedicated clock with an audio-frequency oscillator. If we set this to a tight pulse waveform, there's no reason why it shouldn't act like a clock, but with a far wider range of frequencies. See Figure 5.

If we now increase the trigger speed (ie. the oscillator frequency) and continue to direct the sequencer's output to another oscillator, we might soon obtain a scream of protest at the output. At a trigger frequency of 13000Hz, the scale repeats 1000 times per second. If you have a high-quality synthesizer, you might hear this as another complex tone. Instruments with lesser electronics will

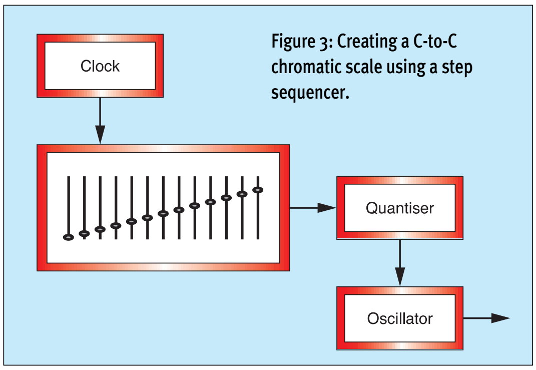


Figure 3: Creating a C-to-C chromatic scale using a step sequencer.

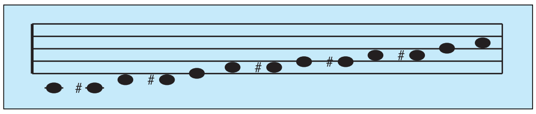


Figure 4: The chromatic scale generated by the configuration in Figure 3.

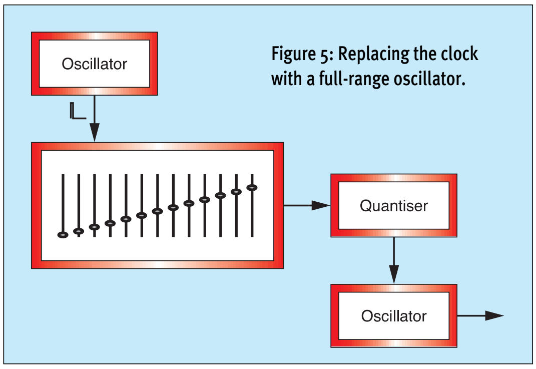


Figure 5: Replacing the clock with a full-range oscillator.

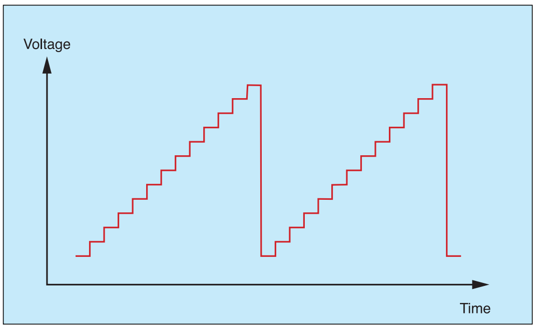


Figure 6: The CV output from the sequencer.

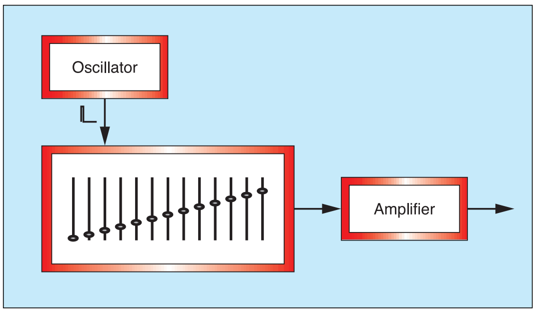


Figure 7: Using the sequencer as a waveform generator.

honour you with silence, perhaps because the clock oscillator can't produce a tight enough pulse at such a high frequency, or because the sequencer can't handle such a rapid trigger, or even because the quantiser can't respond at these speeds.

But let's now look at the quantiser's output in a different way. Figure 6 shows the idealised CV waveform that it produces. It looks a bit familiar, doesn't it? Clearly, it's a 'staircased' version of the common sawtooth wave. So what happens if we take this and, instead of using it to control another module, we treat it as a raw audio signal? The configuration is shown in Figure 7. (For simplicity, I've discarded the quantiser and assumed that the sequencer produces precisely the voltages that we request.)

The result, as you might imagine, is a sawtooth(-ish) waveform, with a frequency of 1/13th that of the trigger oscillator. Indeed, it's even more like a sawtooth than you might imagine because no sequencer can jump instantaneously between voltages, and no amplifier has infinite bandwidth. These limitations smooth the 'steps' in the waveform (see Figure 8). You might say that the output is not really a sawtooth wave, and you would be right — there is a higher proportion of high harmonics than you might otherwise expect. However, it's at least as accurate as that produced by many dedicated oscillators.

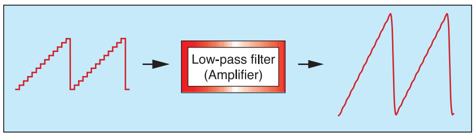


Figure 8: Every amplifier is a low-pass filter.

Let's now return to the idealised situation, and use the sequencer to generate some more waveforms (see Figures 9, 10 and 11). Analysing Figure 9 is easy: it's a 12-step sequence with six steps high and six steps low. Clearly, I'm using the sequencer to generate a square wave. Figure 10 is similarly straightforward: it's a pulse wave with a duty cycle of 1:3. But what about Figure 11? This isn't a waveform that you'll see on a conventional analogue oscillator. It is, however, one that you'll see generated by (hush please... sacrilege alert!) a digital oscillator.

You may not realise it, but we've come to a turning point in our understanding of analogue versus digital synthesis. I'm sure that you had no difficulty coming to terms with the fact that we can perform FM synthesis on an analogue synthesizer (Synth Secrets, *SOS* April and May 2000). Similarly, you probably had no problem accepting that we can perform simple additive synthesis on even a moderately endowed modular analogue (Synth Secrets, *SOS* June 2000). But now I'm going to ask you to accept that we can use our understanding of S&H, slewing, and step sequencers to describe the very fundamentals of digital audio.

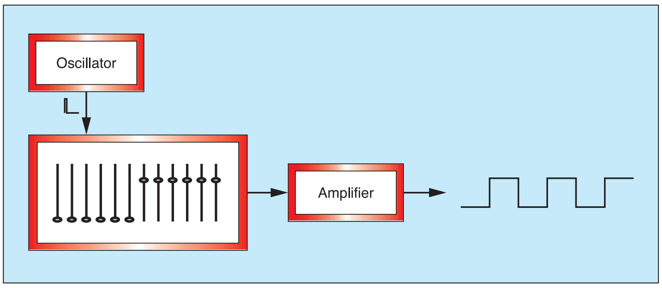


Figure 9: Using the sequencer as a square-wave generator.

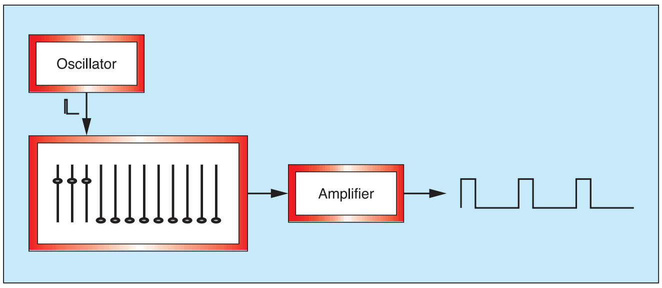


Figure 10: Using the sequencer as a pulse-wave generator.

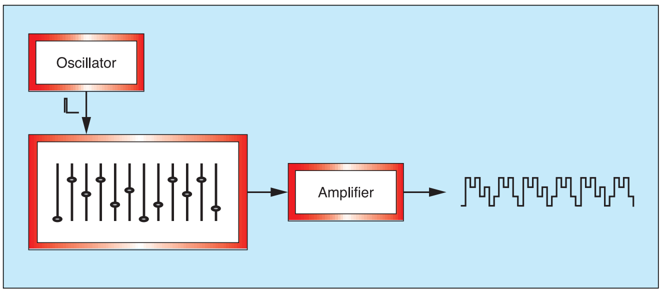


Figure 11: Using the sequencer as a more complex wave generator.

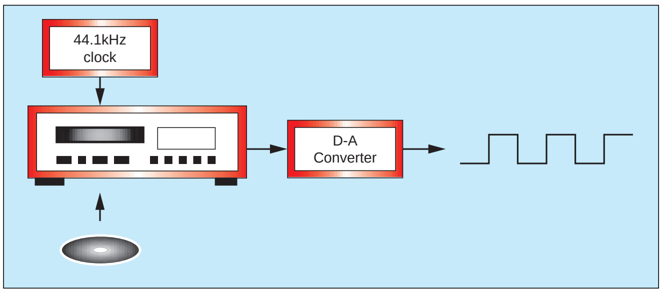


Figure 12: Generating a square wave from a CD.

## Analogue Or Digital?

Figure 9 depicts an ideal square wave generated by a 'perfect' analogue step sequencer. Figure 12 also shows an ideal square wave, but this time it's generated by a CD player reading the digital data held on a CD. To explain why this is relevant, let's remind ourselves about what's happening in the CD player. A bunch of samples are held as binary numbers on the CD itself and, on playback, one sample is read and presented to a digital-to-analogue converter (DAC) each 1/44100th of a second.

If we keep everything simple, and ignore detailed discussions of the coding used in CD technology, we could say that the 'low' part of the square wave is stored as binary 0000000000000000, a 16-bit sample of minimum amplitude. Likewise, we could say that 1111111111111111, a 16-bit sample of maximum amplitude, represents the 'high' part of the square wave. Let's now postulate that we have six 'low' samples followed by six 'high' samples, and that this pattern repeats itself for the duration of the CD. The result is a constant square wave with a frequency of (44100/12) Hz — which is 3675Hz.

Let's now return to the ideal analogue sequencer in Figure 9, and (using a suitably rapid clock) trigger it 44100 times per second. The result is... a constant square wave with a frequency of (44100/12) Hz — which is 3675Hz. Clearly, within the limits of the electronics employed, the results are identical.

Likewise, we can use our CD player to reproduce the waveforms shown in Figures 10 and 11, simply by storing and replaying appropriate samples on a CD. This implies that we can also store and replay synth solos, drum loops, the Vienna Philharmonic playing Beethoven's 6th Symphony... or anything else, provided that we can represent the amplitude of the original signal using a succession of binary numbers. So, where do these 'digital' samples come from?

## Analogue To Digital

Let's return to Synth Secrets 16, and our discussion of Sample and Hold circuits. Figure 13 shows what happens when we sample a sine wave approximately six times every cycle. The output waveform looks very 'blocky', but what is important at this stage is that we have taken a continuous waveform and reduced it to a small number of discrete measurements. We can then pass these measurements to another stage within an analogue-to-digital converter (ADC) that accepts each voltage sample in turn, converts it to a binary number, and sends the number to an appropriate

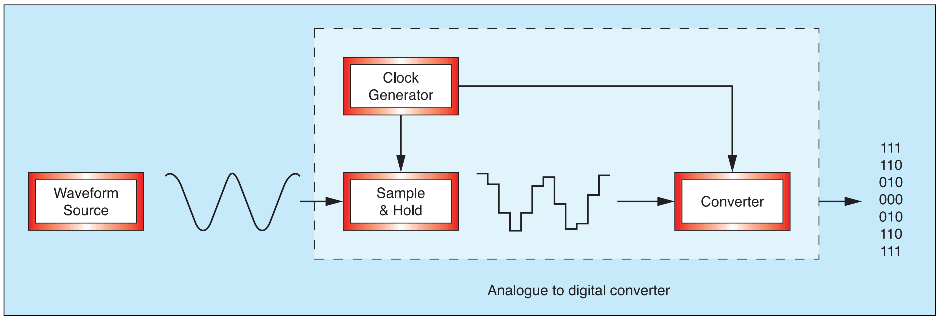


Figure 13: Analogue-to-digital conversion.

storage medium.

Of course, nothing is ever that simple, so we must consider the limitations of the digitising process. Ignoring the inadequacies of real-world electronics, there are two major sources of limitation — one related to amplitude, the other to timing.

Let's look at the first of these. It's simply this: we cannot sample the voltage of the incoming signal with infinite resolution — there is a limit to the number of ways in which we can chop up the amplitude.

If you think that, at this point, we're a million miles from discussions of analogue synthesizers, S&Hs and sequencers, you're wrong. Look at Table 1, parts of which I've copied from last month's Synth Secrets. Previously, I used these values to illustrate voltage quantisation, demonstrating how a quantiser will accept any voltage lying between a given maximum and minimum, and output the central value of that band. Instead of outputting a quantised voltage, the ADC takes any voltage lying between a given maximum and minimum, and outputs a unique binary number that represents the central value of that band (see Figure 14). Sure, last month's context was different, but the two processes are conceptually very similar. The major difference is, perhaps, that last month we *chose* to quantise the levels, whereas this month we have no choice, because digital numbers of finite word-length can only represent a finite number of levels.

| Minimum Input V | Maximum Input V | Output Sample |
|---|---|---|
| 0.958 | 1.042 | 001100 |
| 1.042 | 1.125 | 001101 |
| 1.125 | 1.208 | 001110 |
| 1.208 | 1.292 | 001111 |
| 1.292 | 1.375 | 010000 |
| 1.375 | 1.458 | 010001 |
| 1.458 | 1.542 | 010010 |
| 1.542 | 1.625 | 010011 |
| 1.625 | 1.708 | 010100 |
| 1.708 | 1.792 | 010101 |
| 1.792 | 1.875 | 010110 |
| 1.875 | 1.958 | 010111 |
| 1.958 | 2.042 | 011000 |
| 2.042 | 2.125 | 011001 |
| 2.125 | 2.208 | 011010 |
| 2.208 | 2.292 | 011011 |
| 2.292 | 2.375 | 011100 |
| 2.375 | 2.458 | 011101 |
| 2.458 | 2.542 | 011110 |
| 2.542 | 2.625 | 011111 |
| 2.625 | 2.708 | 100000 |
| 2.708 | 2.792 | 100001 |
| 2.792 | 2.875 | 100010 |
| 2.875 | 2.958 | 100011 |
| 2.958 | 3.042 | 100100 |

Table 1: Digitising a range of analogue voltages.

The second limitation in analogue-to-digital conversion concerns the sampling frequency of the system: just as we can not digitise the amplitude with infinite precision, we can not conduct the measurements with infinite speed.

We now arrive at the most important theorem in digital audio. This is the Shannon-Nyquist Sampling Theorem, named in honour of the two gentlemen who demonstrated and proved its veracity. (The Theorem is not specific to digital audio, but applies to any sampled signal.) You'll be pleased to know that proving the Theorem is so far from trivial that I won't even attempt to show it here. However, I can easily state its most important tenet, as follows:

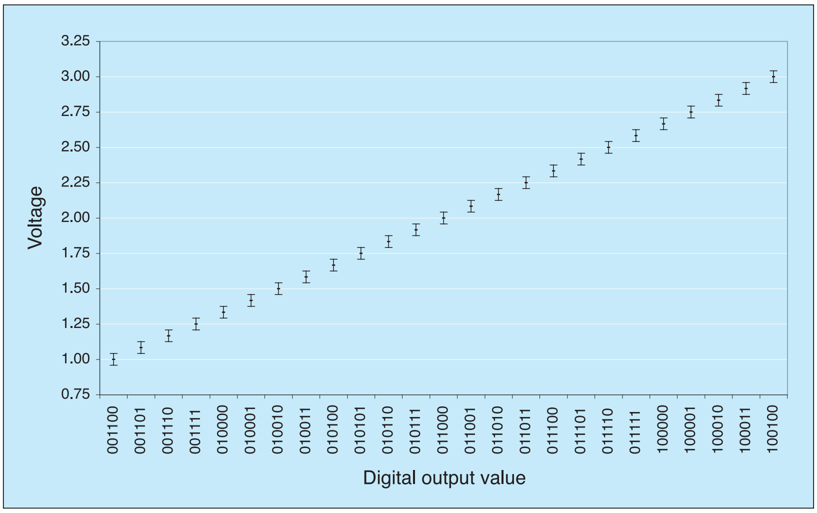


Figure 14: In an analogue-to-digital converter, every sampled voltage produces a binary number.

*If a signal has a bandwidth of less than F (the Nyquist frequency), then 2F samples per second are sufficient to represent that signal fully and unambiguously.*

There are a couple of hidden conditions in this statement. (The first is that a limited bandwidth implies a signal of infinite duration, while the second is that you must sample the analogue signal at evenly spaced intervals.) But we are not going to worry about these. Instead, we're going to look at its most important consequence:

*An analogue signal can be reconstructed, without error, from samples taken at equal time intervals, provided that the sampling rate is greater than twice the highest-frequency component in the original signal.*

It's possible to demonstrate this pictorially. Look at Figure 15. This shows what happens when you sample a sine wave at exactly five times its frequency. If, for the sake of argument, the waveform has a frequency of 10kHz, this represents a sampling rate of 50kHz, and it should be obvious that this is more than sufficient to describe the sine wave accurately. Figure 16 depicts the same situation, except that I have reduced the sampling rate to 25kHz. As you can see, the samples are more widely spaced, but they remain sufficient to define the waveform unambiguously.

Figure 17 demonstrates a completely different state of affairs. The samples still lie on the input signal, but they now depict a waveform (the orange line) that completes one cycle for every three cycles of the original. This means that there is, in principle, a different frequency — 3.33kHz — in the sampled data. How has this happened?

The answer lies in an effect called 'aliasing'. I'm sure that you have encountered this name before, especially if you own an FM synthesizer, but maybe you've never known exactly what it is. So here's the explanation. The sample rate in Figure 17 is 13.33kHz, and the Nyquist Theorem tells us that the maximum frequency that this can accurately represent is 6.67kHz. Since this sample rate cannot handle our 10kHz input, a strange effect occurs: any frequencies above the Nyquist frequency 'fold over' and appear an equal distance below the Nyquist frequency. The high frequencies reappear 'under an alias'. In this example:

- The Nyquist frequency = 6.67kHz.
- The input signal = 10kHz.
- The 'aliased' frequency is (10kHz minus 6.67kHz, subtracted from 6.67kHz) = 3.33kHz.

Figure 18 makes this effect even more obvious. This shows our 10kHz source sampled at just 11.11kHz, and you can see clearly that the samples, which still lie on the source waveform, now describe a wave of 1.11kHz.

Unfortunately, you can't remove aliasing once it has been introduced, so we have to ensure that it never occurs. We do this by making sure that there are no frequencies above 'F' in the analogue signal presented to the ADC. How? By sticking a low-pass filter in the signal path. However, this filter introduces more problems — in particular, the detrimental phase shifts described in Synth Secrets 4 (*SOS*, August 1999). When you analyse it, all this digital stuff is very analogue!

## Digital To Analogue

Once we've digitised our audio, we still need to convert it back to analogue in order to hear it. So, how do we get back to the original using our samples as a starting point? Let's forget all about audio — analogue or digital — for a moment, and return to the schoolroom. It's 2:30pm and you're stuck in double maths. Yurg! Up at the front of the class, your teacher has drawn a curve, together with the common graphical representation for a set of seven samples taken along its length (see Figure 19).

Given these samples, I'm going to tell you that I know the exact amplitude of the curve at every moment along the 'time' axis. How? Well, look at Table 2, which shows the values of the samples in Figure 19.

If you think back to your school maths, you'll recognise what I've done. The curve is a parabola, with the amplitude of any sample equal to the square of the Time. We write this as Equation 1.

It looks simple, and it is. Nevertheless, it hides a marvellous secret. Whereas the samples in Figure 19 were 'taken' at integer times (2, 5, 8, and so on) the equation allows us to calculate the amplitude at any time we like within the span of the given samples. This could be another integer time, but it could also be at T = 3.6347563356, or whenever else we choose.

At this point, the cynical among you might say, "OK, the line goes through the points, and it could

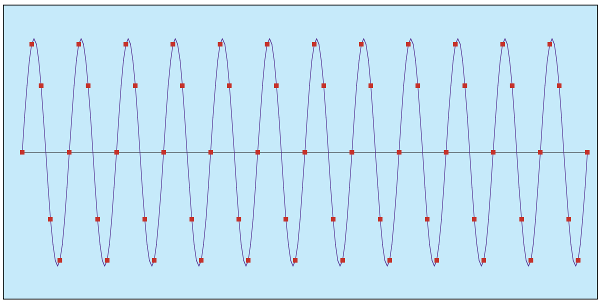


Figure 15: Sampling a 10kHz sine wave at a 50kHz sample rate.

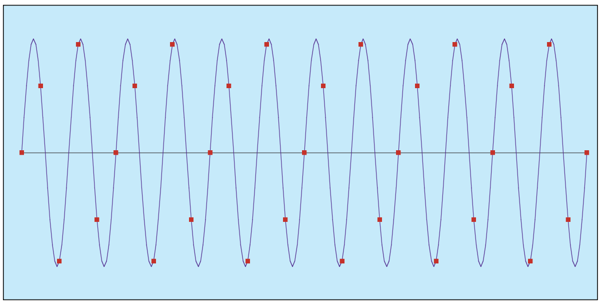


Figure 16: Sampling a 10kHz sine wave at a 25kHz sample rate.

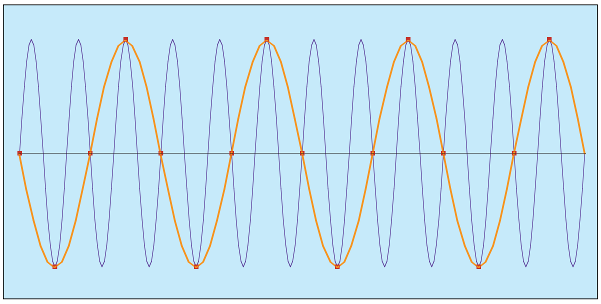


Figure 17: Sampling a 10kHz sine wave at a 13.33kHz sample rate.

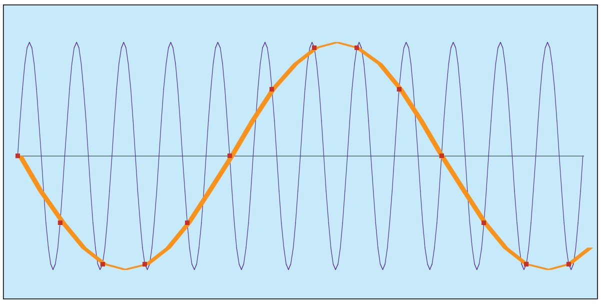


Figure 18: Sampling a 10kHz sine wave at an 11.11kHz sample rate.

$$S = T^2$$

Equation 1: The equation for a parabola, where S = the sample amplitude and T = the time.

be a parabola, but why can't it be something more complex and 'wiggly' — a cubic equation, or something more complex than that? Wouldn't that screw your argument completely?" Well, yes, it would. But I'm going to play God and set a limit on the complexity of the signal: it is only allowed to be a parabola. Then there will only be one line that can possibly fit the available samples.

Now let's return to the real, audio world and ask, "Given a set of samples derived from an audio signal, can we postulate a limiting condition that allows us to fit one, and only one, waveform to those samples?" Amazingly, the answer is "yes". Remember: if the sampling rate is greater than twice the highest-frequency component in the original signal, it is sufficient to represent that signal fully and unambiguously. The only condition required is that the bandwidth is limited to less than half the sample rate! Given this knowledge, we can design a 'reconstruction filter' (which is another form of low-pass filter) that exactly reproduces the original analogue signal from the sampled data.

(Given the 'stepped' representation of samples in Figure 19, it's not hard to see why so many people are fooled by ignorant "audio stored digitally sounds horrible because it's a series of steps, not like real music" bullshit. But that's exactly what it is: bullshit. Digital audio might sound horrible because it's badly implemented, but that's another story.)

| Time | Sample Value |
|---|---|
| | |
| 0 | |
| 1 | |
| 2 | 4 |
| 3 | |
| 4 | |
| 5 | 25 |
| 6 | |
| 7 | |
| 8 | 64 |
| 9 | |
| 10 | |
| 11 | 121 |
| 12 | |
| 13 | |
| 14 | 196 |
| 15 | |
| 16 | |
| 17 | 289 |
| 18 | |
| 19 | |
| 20 | 400 |

Table 2: The values of a sampled parabolic waveform show in Figure 19.

## A Synth Secret Or Something More?

The ideas in this month's Synth Secrets form the basis of all digital audio, and make possible CDs, DATs, samplers, digital effects units, drum machines, digital audio workstations, and everything else in the digital audio studio. Furthermore, they mean that we're now in a position to answer the question I asked in the first paragraph of this article. To whit: is the red line in Figure 1 the input from an S&H module, with the blue line the smoothed output? Alternatively, is the blue line an audio signal input, with the red line the S&H output?

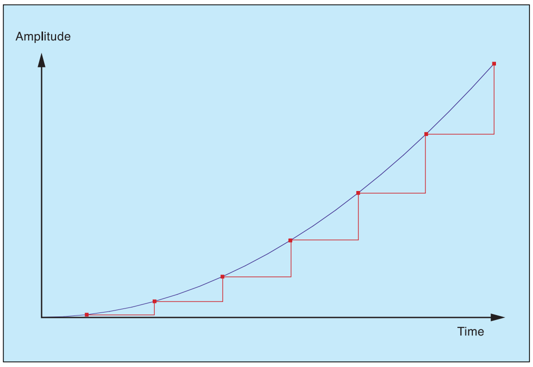


Figure 19: A snippet of sampled waveform.

Indeed, let's extend the question: is the blue line in Figure 1 an analogue signal, with the red line the output from an analogue-to-digital converter? Or is the red line the signal stored on a CD/hard disk/whatever, and the blue line the analogue output generated by a digital-to-analogue converter?

Thanks to our understanding of S&H modules, slew generators, step sequencers, and low-pass filters, we now know that the answer can be "all of the above." As I've said many times before, I love this stuff! SOS

# synth secrets

## PART 18: PRIORITIES & TRIGGERS

In these days of 64-note polyphony and 32-part multitimbrality, it's easy to forget the importance of note-priority systems in analogue monosynths — yet they can have a drastic effect on what you hear when you play or trigger an old synth. **Gordon Reid** provides a refresher course.

For 17 months we've been talking about the nature of sounds and how we might represent them using simple analogue electronics. In a few months, we're going to start using this knowledge to dissect real sounds and recreate them using the components of typical analogue synthesizers. But before we get carried away by all the maths and physics, we should remember that synths are, above all else, musical instruments. This means that, before we can get the best from them, we must understand and become proficient at the ways in which we can control them.

Lest this sound like a clarion call to keyboard-playing snobs everywhere, it's worth pointing out that in the context of the typical *SOS* readers' setup, the way in which you 'control' a synth might include playing a guitar and using some form of guitar synthesis, using vocal and/or other control signals to activate a synth using pitch/CV converters and envelope followers, or using some other form of external control, such as a sequencer. It could include waving your hands in the air in front of a proximity- or light-sensitive controller, or tweaking the resonance and filter cutoff knobs of a Roland TB303 whilst the internal sequencer cycles around a simple note pattern. Despite all this, it doesn't take much to point out that the primary method of controlling most analogue synths *is* via a keyboard, which is why I will be using this interface to illustrate most of the points that I make this month.

That said, here's a Synth Secret that I regard as axiomatic:

*If we're going to get the best from our synthesizers, we need to understand how their sounds react when played — whatever means we use to play them.*

This no-brainer was brought home to me a few weeks ago when I unearthed and started playing one of my lesser-used analogue monosynths — a Yamaha CS20 — with unexpectedly unpleasant results. If you had been there, you might have heard me muttering "the old girl's up the Swanee" (or something less printable) in dark tones. Half the notes seemed to be speaking a fraction of a second after I played them, and for a few moments, I was convinced that the trigger circuit was misbehaving. Fortunately, there was nothing wrong with my synth; I had simply forgotten that the CS20 has a high-note priority keyboard. I, being an ARP man at heart, am a low-note priority sort of guy, so I wasn't releasing the upper notes quickly enough, thus making the low ones speak too late. If all this means nothing to you, then you should *definitely* read on.

Minimoog — an example of a lowest-note priority synth...

...and the Korg 700S — an example of a Japanese, highest-note priority synth.

Photo courtesy of Korg Japan.

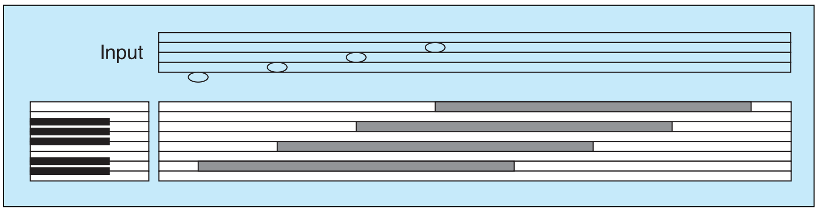


Figure 1: A simple upward sequence played polyphonically.

## A Matter Of Priorities

You would think that there's very little to complicate using a monosynth, wouldn't you? You just set up a sound on the control panel, press a note or send it a control voltage if it has an external input, and everything responds exactly as you tell it to — right? Well, this may be true in isolation, but not when you start to play more than one note at a time.

Hang on a second... we're discussing *monophonic* synthesis. Surely, there's never a time when you're going to play two or more notes simultaneously, is there? Actually, in all likelihood, you press two notes simultaneously every time you play a line on your synth. Whether you're playing a simple melody, a bass riff, an accompaniment, or an Emerson-esque lead extravaganza, it's unlikely that you'll release every note before playing the next. If you try, you'll probably play very unevenly, in a disjointed staccato fashion. So what effect do these little overlaps have?

Look at Figure 1 (left). This shows two representations for the simple four-note sequence D-F-A-C. Each note lasts for four beats, but the start of each note is separated from the previous by one beat. OK, so the notation in the upper stave is not strictly according to the rules (there should be a key signature, rests and so on) but we're not going to worry about that. Furthermore, the bar lasts for seven beats (which puts it in 7/4) but I'm an old prog-rocker, so we're not going to give that a second thought, either.

Now look at the lower representation of the sequence. This is my personal shorthand for depicting the pitch of the note, when it starts, and how long it lasts. It's very similar to the grid edit pages of some sequencers and drum programs, so I imagine that you'll have seen something similar elsewhere. As you can see, the first note starts on beat one and lasts until beat five. The second starts on beat two and lasts until beat six... and so on.

If we play this sequence (which I'll call the 'input') on an organ or any other truly polyphonic keyboard, we hear the four notes speak on successive beats. After that, we then hear them ▶

ending on successive beats after the fourth note has spoken.

Unfortunately, and by definition, a monophonic synthesizer can only produce one note at a time. Fortunately, at some point during its design, someone will have decided how it will respond to this situation.

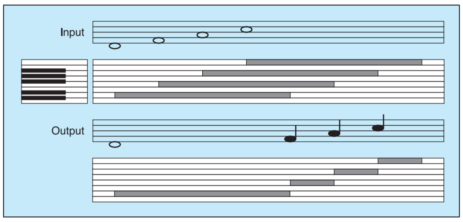


Figure 2(a): The sequence from Figure 1 played on a lowest-note-priority monosynth.

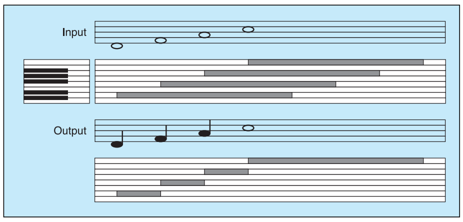


Figure 2(b): The same sequence played on an highest-note-priority monosynth.

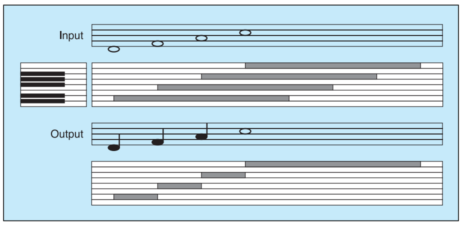


Figure 2(c): The sequence played on a last-note-priority monosynth.

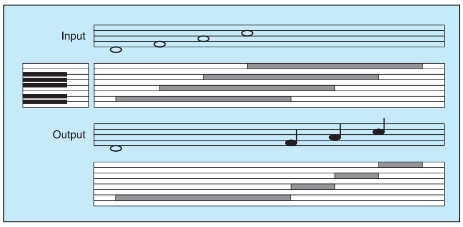


Figure 2(d): The sequence played on a first-note-priority monosynth.

The most common response is called ‘lowest-note priority’. This means that the lowest note held at any time is the one that the synthesizer will produce. That might sound simple, but Figure 2(a) shows what might happen to our sequence when played on a lowest-note-priority monosynth. As you can see, the output is quite different from the input. Because it’s the lowest note, the first note is held for its entire duration, and the following notes can only speak when they become the lowest held notes.

The other common response is ‘highest-note priority’. In direct contrast to lowest-note priority, this means that the highest note depressed at any time is the one that the synthesizer will play. Figure 2(b) shows what will happen to our sequence when played on a highest-note-priority monosynth. At least the notes speak at the right time with this approach, but they are curtailed when you play the higher ones. But before you suggest making all monosynths highest-note priority, I’m afraid that it isn’t a panacea (as we’ll see in a moment).

Before explaining why not, let’s look at two less common priority systems. The first of these is ‘last-note priority’. Figure 2(c) shows the same sequence replayed using a monosynth based on this system. As you can see, the last (ie. most recent) note has priority, so you always hear the last note you pressed. Of course, this looks identical to the results you get with highest-note priority. But bear with me a moment.

The last of the four schemes is ‘first-note priority’. In this case the first (rather than the most recent) note has priority. Figure 2(d) shows the result. This time, the output looks identical to that of the lowest-note-priority synthesizer.

So, can we say that last-note priority is the same as highest-note priority, and first-note the same as lowest-note? Sorry, but in general, this is *not* true. It only seems like that because I chose an upward sequence of four notes to illustrate the problem.

So let’s look at the converse of this, and take a simple *downward* sequence. Figures 3(a), 3(b), 3(c), and 3(d) show the same four priority systems, but the results are quite different from before.

With this sequence, lowest-note priority offers the same results as last-note priority, while first-note priority imitates highest-note priority. This is opposite to the result we achieved when playing up the keyboard, and I hope that the diagrams make it obvious why! Furthermore, Figure 3(b) explains why I said that highest-note priority is no panacea... it only lets notes speak on time when you play up the keyboard.

Let’s now complicate matters by playing a four-note sequence that changes direction. Study Figures 4(a), 4(b), 4(d), and 4(d), which show the input and outputs for such a sequence (see page 182). As you can see, the results are quite unexpected, and each is different from the other. Indeed, two of the sequences play just three notes, and the timing can be very odd compared with the original, regular beat.

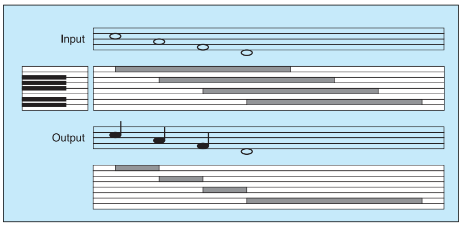


Figure 3(a): A downward sequence played on a lowest-note-priority monosynth.

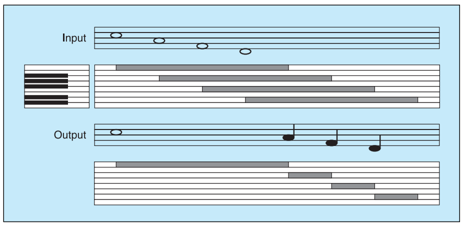


Figure 3(b): The same sequence played on a highest-note-priority monosynth.

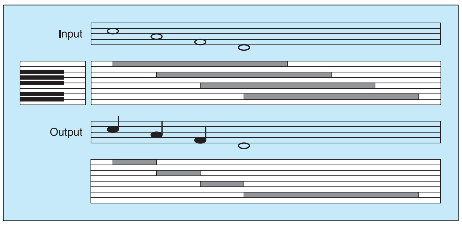


Figure 3(c): The downward sequence played on a last-note-priority monosynth.

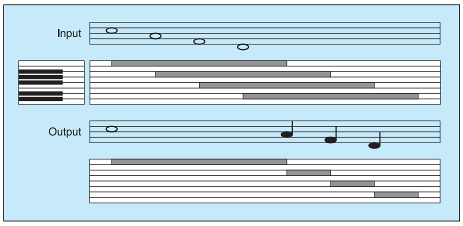


Figure 3(d): The downward sequence played on a first-note-priority monosynth.

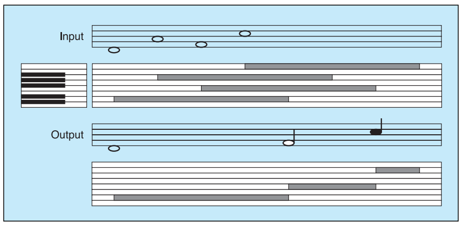


Figure 4(a): The mixed sequence played on a lowest-note-priority monosynth.

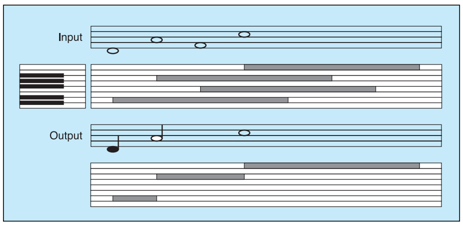


Figure 4(b): The mixed sequence played on a highest-note-priority monosynth.

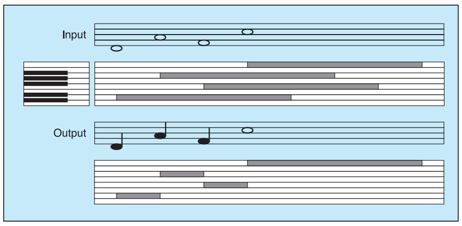


Figure 4(c): The mixed sequence played on a last-note-priority monosynth.

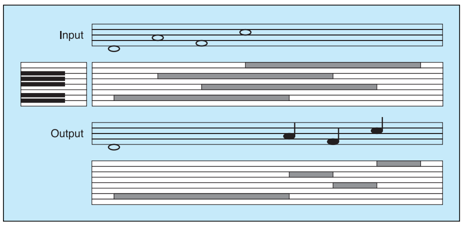


Figure 4(d): The mixed sequence played on a first-note-priority monosynth.

▶ At this point, I could continue to draw increasingly complex sequences, and show you how the four priority schemes make a mess of your playing. But I won't, because I suspect that you could now do this for yourselves without any further help from me!

## The Real World

Of course, Figures 2(a) to 4(d) are extreme examples. It's highly unlikely that you will hang on to a note for three full beats after you intended to release it, but the principle remains the same, no matter how long or short the overlap. So here are a couple of rules to remember:

*(i): Hang on to a lower note for too long on a lowest-note-priority monosynth, and the following higher note will speak too late, making your playing sound uneven and off the beat. The same is true if you hang on to a higher note too long on a highest-note-priority monosynth, and then follow it with a lower note.*

*(ii): Hang on to a lower note too long on a highest-note-priority monosynth, and the following higher note will still speak on time, so you'll get away with your bad technique. The same is true if you hang on to a higher note too long on a lowest-note-priority monosynth, and follow it with a lower note.*

As you can see, I've limited these 'rules' to lowest- and highest-note-priority monosynths because these are by far the most common, and therefore the ones that you're most likely to encounter. However, since all keyboard playing involves playing up and down the keyboard, no matter which synth you're playing, you'll trip over these problems if you hold notes too long. The only exception to this will be a last-note-priority monosynth which, in general, will always play the notes at the moment that you press the keys.

## Triggers

Up to this point, we've treated synthesizers as monophonic organs, and ignored the actions of their contour generators and the triggers that initiate their actions. But we can't carry on doing this indefinitely. To illustrate this, let's take the lowest-note-priority synth in Figure 4(a) as an example. In this case, the synthesizer holds the first note for its entire four-beat duration because this is the lowest note in the sequence. When this note expires, the synth then plays the remaining beat of the second note followed by the third note and, finally, the fourth note. But to do this, it must still be producing a sound at beats 4, 5, 6 and 7. If the contour generators have already closed the VCF, or reduced the gain of the VCA to zero, no sound will be forthcoming whether you play another note or not — see Figure 5.

To overcome this, we must cast our minds back to part seven of this series (see *SOS* November '99) and turn our thoughts once again to the triggers that initiate the contour generators which shape the loudness and tone of our sounds (see Figure 6). However, we must now extend these ideas to encompass keyboard priorities as well as the triggers and contours we discussed months ago.

As you know, the modules in Figure 6 need something to produce the trigger that initiates the contour generators, and to tell the VCO what pitch to produce. Bearing in mind what I said at the start of this article, let's simplify things and say that for the purposes of discussion here, the 'something' required to initiate the trigger is the key that you press on the synthesizer's keyboard.

Now let's illustrate this further. Figure 7 shows a simple keyboard synthesizer with a single VCO, a single contour generator, and a single VCA. It has no filter, so it can't shape the tone of the note, but it's perfectly adequate for our discussion. As you can see, the keyboard produces three output voltages. These are: a pitch CV to

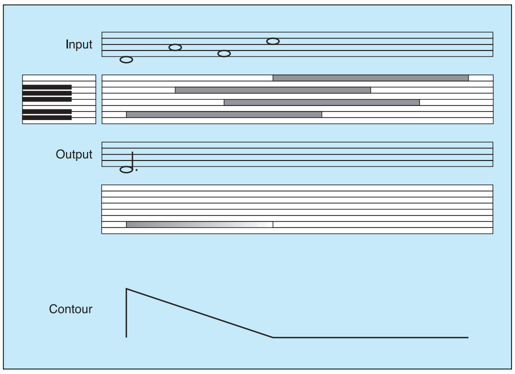


Figure 5: A sequence played on an AR-voiced, lowest-note-priority monosynth.

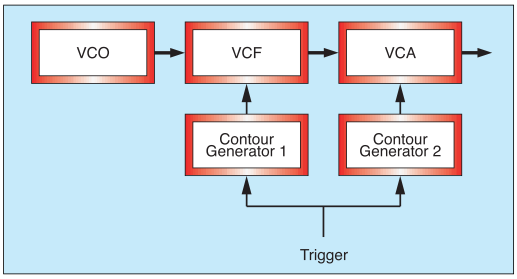


Figure 6: A simple VCO/VCF/VCA monosynth.

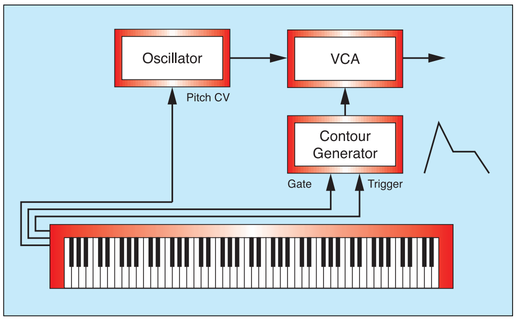


Figure 7: A simple keyboard synthesizer with a VCO, envelope and a single VCA.

determine the pitch of the note; a Trigger to tell the contour generator when to 'fire'; and a Gate that tells the contour generator how long you keep a key (or keys) depressed.

Now let's apply the ideas contained in Figure 7 to the rising scale in Figure 2(a). Can you see the problems? For example, does the keyboard send a trigger only when all existing notes are released and a new key is pressed in isolation (so-called 'single triggering')? If so, do the contour generators allow sound to pass for the whole duration of the Gate? Alternatively, does the keyboard generate a trigger whenever you press a key, irrespective of whether existing notes are depressed or not (so-called 'multi-triggering')?

Let's look at the first of these cases applied to a lowest-note-priority keyboard. Figure 8 shows that, provided that the sustain is non-zero for the duration of the Gate, all the notes play, albeit at the wrong times, and without the correct articulation of the individual notes (this, by the way, is exactly how a Minimoog works).

In contrast, Figure 9 (which depicts the lowest-note-priority multi-triggering of an ARP synth) is nasty, because it shows that all four articulations occur during the first note, not when the other notes finally speak (this is where we diverge from what we learned in part seven of this series, which assumed last-note priority throughout). OK, so I've grossly exaggerated the extension of each of the notes, but the problem can become severe, even when you drag your notes just a little. Where you should have a release phase of the previous note and then a nice, sparkling, contoured new note, the last few moments of the old note are affected by the attack phase of the next contour. This can sound very unnatural, and is it a common performance error.

Indeed, further extending the ideas mentioned in part seven, I can make Figure 9 look and sound even worse by replacing its contour generator with one that 'resets to zero' each time it receives a new trigger (see Figure 10). The sustained D is now chopped up into sections, with no articulation of the later notes. Horrid!

But this still isn't the strangest trigger characteristic. It's not uncommon to find a synth that retriggers when you release previous notes, provided that at least one note is still depressed (see Figure 11). This can be quite useful, because it makes sounds speak correctly when you release the previous notes, but it makes slurs impossible because, in general, you can't make notes run into each other.

## Putting It All Together

So where does this leave us? We have four pitch-priority schemes, and six or more permutations of triggering/contouring. This means that there are at least 24 keyboard characteristics that you might encounter when playing a monophonic synthesizer, each of which can demand a slightly different playing style if you are to achieve optimal results.

You could, I suppose, take the simplistic view and say that, if you play perfectly or sequence carefully, none of the above limitations will arise. But you can also turn this view around, and put some of the 'limitations' of these priority systems to your advantage. Here's an example of a technique that was used by numerous soloists in the '70s.

Using a multi-triggering, highest-note-priority synth, hold a low note (say, a bottom C) using one of the fingers of your left hand or, if you prefer, a wooden stake, a Hitler Youth dagger, or something similar (fans of Keith Emerson will know what I mean). Then play a solo using your right hand. If you play legato higher up the scale, the synth will perform conventionally and each note will be articulated correctly. But if you cease playing, the pitch will drop almost instantaneously to the bottom C and, if the VCF and VCA sustains are greater than zero, continue to produce this note until you start playing again.

You can develop this technique still further. Consider this: when you let the pitch drop to bottom C, you have not pressed a key (you have only released one) so — on most synths — the low note will not be articulated. This can be an expressive and subtle performance feature. If you now add a little portamento to the patch, the synth will swoop down without articulation, giving the note a different character from the main body of the solo. Then, when you press another key to continue playing, a trigger occurs, the contour generators do their stuff, and the pitch swoops up in a flurry of increasing amplitude and opening filters.

Many players used this technique to

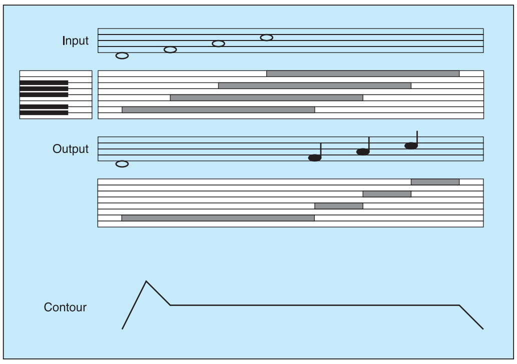


Figure 8: A 'single-trigger' synth retriggers only when you release all other notes.

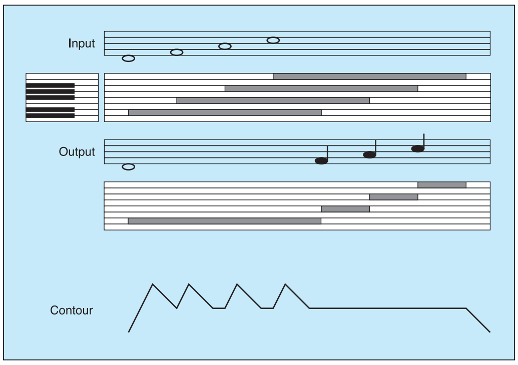


Figure 9: A 'multi-trigger' synth retriggers every time that you press a key, but does so in the 'wrong' place.

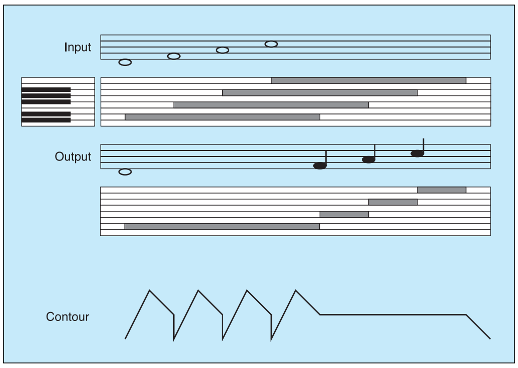


Figure 10: A 'reset-to-zero' contour chops up the note that is held for too long.

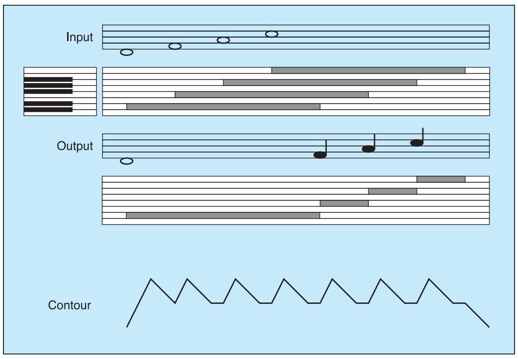


Figure 11: Some synthesizers retrigger on any transition between notes.

**“Played as a rapid staccato with just a touch of portamento, and preferably with a huge Mellotron chord under your left hand, the result makes you sound like a virtuoso even though you can achieve it using just one finger of your right hand.”**

play impossible solos spanning two or three octaves. They did this by leaving short gaps between each of the notes played by their right hands, thus causing the bottom note to play between each. Played as a rapid staccato with just a touch of portamento, and preferably with a huge Mellotron chord under your left hand, the result is extremely distinctive. It makes you sound like a virtuoso (well, actually, it makes you sound like Rick Wakeman) even though you can achieve it using just one finger of your right hand.

## The Real World

I thought that it would be fun to finish this article by taking a short tour of some common monosynths to study their keyboard priorities and triggering regimes. So let’s start with the most famous of all — the Minimoog. This, as you would expect from the world’s first integrated synthesizer, is one of the simplest of the bunch and, as stated above, uses lowest-note priority and single triggering. The same is true of the Minimoog’s younger and less well-endowed little brother, the Prodigy. ARPs such as the Axxe and Explorer also use a lowest-note priority system, but retrigger when you play a higher note — even though you still hear the pitch of the lower! Indeed, my Axxe often (but not always) retriggers when you release higher notes, too. I suspect that this is a fault, but I’m not complaining.

Jumping across to Japan, the Roland SH09 is another single-trigger keyboard, but with highest-note priority. The same is true of the Korg 700S, the Korg MS20, and the Yamaha CS20 which, of course, is where this article began. However, other Rolands respond differently. For example, the SH2000 has highest-note priority and retriggers both on pressing and releasing any key (except the last).

I should also mention the Crumar Spirit. This boasts a last-note-priority system with a unique twist... It remembers the first note that you play and, once you have released all subsequent notes, it returns to this (provided, of course, that you are still holding it). This means that its keyboard combines last-note and first-note priorities! It may sound crazy, but it makes the Spirit an almost uniquely playable instrument.

So there you have it... even if you ignore the sound generation itself, not all monosynths are equal. Indeed, it seems that the Pacific Ocean determines how you play your keyboards. American synths are, almost without exception, lowest-note priority, while Japanese ones are predominantly highest-note priority.

This provokes an interesting thought. Since the heyday of the American monosynths was from 1970 to 1975, and the Japanese manufacturers arguably took over from 1976 to 1981, it’s also true to say that highest-note priority overtook lowest-note priority as the dominant system. Maybe this is another reason why players who cut their teeth on Moogs and ARPs disliked later monosynths. In music, as in all else, we’re always happiest with the system we learned first.

Consequently, when you choose a monosynth, you should ask yourself whether you prefer lowest-note, highest-note, or last-note priority (these being the three that you are likely to encounter) and whether you prefer single- or multi-triggering. These serious considerations could prove as important to you as the action of a guitar, or the placement of the individual drums in a kit.

Next month, we’ll start looking at duophonic keyboards and beyond. Until then, have fun prioritising. SOS

# synth secrets

## PART 19: DUOPHONY

**Gordon Reid** discovers that two's company, as he investigates how manufacturers stretched the capabilities of analogue monosynths to offer the magnificent total of two notes at a time...

When is a monosynth not a monosynth? The answer, of course, is when it's a duosynth and can play two notes at a time rather than just one. No big deal, you might think.

But is the synth in question merely duophonic or does it offer a more powerful duo-timbral architecture? Does it offer single triggering or multi-triggering, and are there different ways in which you can combine the sounds generated? There's more to this duophonic malarkey than you might at first imagine.

### The Birth Of Duophony

Let's start at the beginning. With few exceptions, early analogue synthesizers played one note at a time. It didn't matter how many oscillators, filters, contour generators or whatnots you threw into a sound, they all responded together when you played a key on the controlling keyboard. This is not surprising, as it's relatively simple to design a keyboard that generates a single pitch CV, a single trigger and a single gate.

Figure 1 represents a typical monosynth. It has two oscillators whose pitches are controlled by a common pitch CV, determined by the choice of key that you play on the keyboard. You can select the waveforms generated by each oscillator, and you can even detune one with respect to the other, but the single pitch CV ensures that the relationship between them remains constant, no matter which key you play. (This being a monosynth, it doesn't matter whether you play more than one note at a time. Only one will produce a pitch CV — see last month's instalment of 'Synth Secrets' for more details.)

The outputs from the two oscillators are mixed before they travel along the rest of the signal path, to a VCF and a VCA whose actions are controlled by a pair of contour generators. These can respond quite differently from each other (ie. you can set different values for the contour stages) but they are always initiated by the same trigger and gate pulses generated when you press the keys. Given this configuration, we need only one item of information — the note priority system — to enable us to predict, with complete certainty, the response of the synthesizer to any notes we choose to play.

Now let's shake things up a bit, and consider what happens if we make it possible for the keyboard to generate two pitch CVs simultaneously. But hang on a moment; before predicting how this will work, we must decide which two notes are going to generate the pitch CVs.

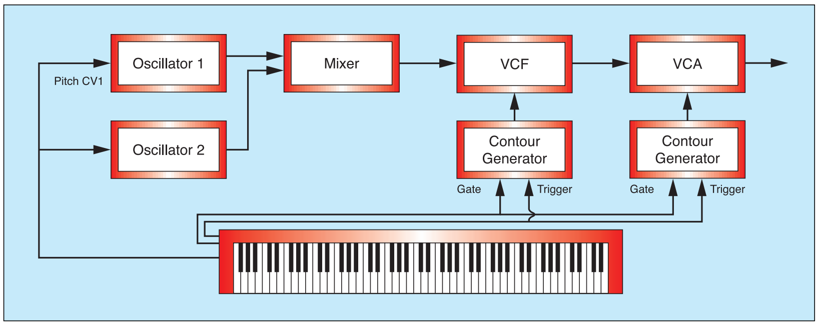


Figure 1: A dual-oscillator monosynth.

Imagine that you're standing in front of a duophonic synth (one whose keyboard generates two pitch CVs) which you have patched to produce a continuous sound, such as an organ. Now play a simple Dm7 chord: D, F, A, and C. Which two notes will you hear? The answer is determined by engineering: it's simpler to design a keyboard that detects the highest and lowest notes than it is to design one that recognises the middle two notes, the most recent pair, or some other combination. Consequently, you will hear the D and the C.

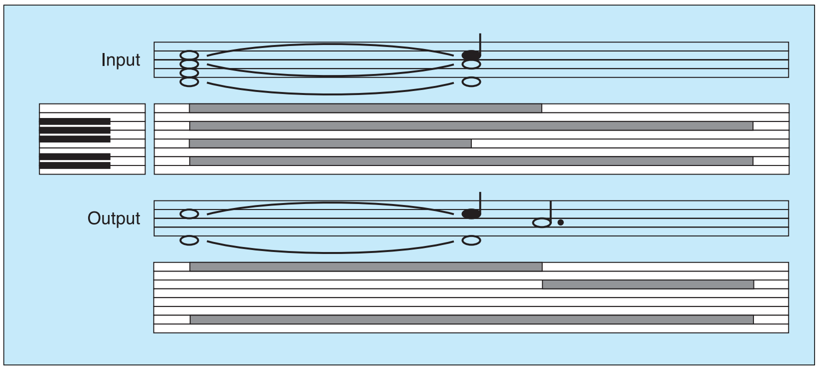


Figure 2: An example of duophonic response.

Now ask yourself what happens when you release the F. Simple. Nothing happens. The D and the C are still the lowest and highest notes (respectively). Now release the C. At this point, the A becomes the highest note, so you hear the D and the A. (See Figure 2.)

ARP's Odyssey was the first commercial synth to offer duophonic capability.

To modify the dual-oscillator monosynth in Figure 1 to give the duophonic response shown in Figure 2, we add a second pitch CV to create Figure 3. With this architecture, the keyboard is capable of generating two independent pitch CVs, with CV1 directed to oscillator 1, and CV2 directed to oscillator 2. Now, when you play the sequence in Figure 2, CV2 (which, for the sake of argument, represents the lowest note) causes oscillator 2 to play the continuous D, while CV1 (representing the highest note) plays the high C followed by the A.

Unfortunately, this explanation leaves two important questions unanswered. Firstly, what happens if you only play a single note? To answer this, we need to jump into the *Sound On Sound* time machine and set the controls for 1972...

## CVs & The ARP Odyssey

The ARP Odyssey was big news when it was launched, partly because it sounded great, and partly because it had a stunning features-per-kilo ratio. But it was something else that made the Odyssey special: it was the first commercial synth that allowed you to play two notes simultaneously. Indeed, the architecture employed by the Odyssey was similar to that shown in Figure 3. OK, it had an LFO, a ring modulator, assignable contour generators, and loads more, but the basics were as shown.

Unsurprisingly, when you play two notes on an Odyssey, it behaves as we have discussed. You can manipulate the oscillators' settings individually, but their outputs are then mixed, and the filters and amplifier modify the composite, duo-pitched sound. Now let's release one of the notes and see what happens. Hey... rather than disappearing (as would happen on a modern polyphonic synthesizer) the second note layers itself on top of the first! This, of course, is why the Odyssey is most often treated as a monosynth; if you play it as one, it responds as one, with both oscillators providing components in a single ▶

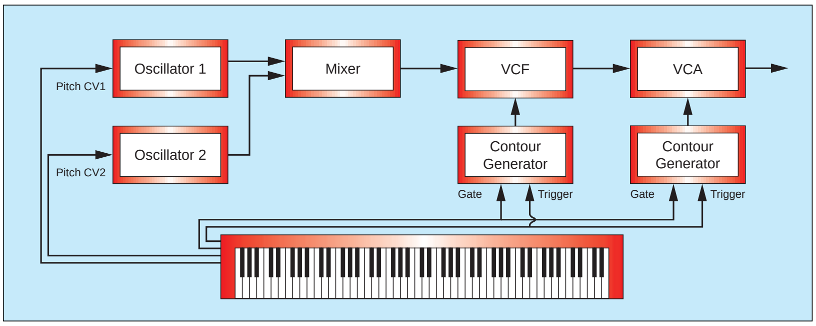


Figure 3: Adding a second pitch CV to the synthesizer in Figure 1.

sound. Of course, what's actually happening here is that the highest note and the lowest note are the same, and jumping from two single-oscillator notes to one dual-oscillator note results in an unnatural change in the tone.

Figure 4 shows how unpleasant this can be. It depicts a single drone held as the lowest note, plus a regular, beating highest note with gaps between each key release and the subsequent key press. (The 'squiggle' between each successive C in the input is called a rest, because that's what it's telling you to do.)

As you can see, when two notes are depressed, you hear them as separate notes of a certain timbre (which I have shown as red and blue). However, when just one note is held (in this case, the lower one) the two oscillators combine to produce a single note of a 'fatter' timbre, shown in purple. What's more, the transitions between these states — shown as vertical arrows — are not instantaneous, and result in a rather unmusical glitch.

As you can see, duophonic synths are more limited, and less intuitive, than you might expect, so here's a Synth Secret:

*Duophonic synthesizers are far less than polyphonic synthesizers restricted to playing two notes at a time.*

## Duophonic Triggering

Let's now consider the second question I promised: how does the triggering regime affect the two-note capabilities of a duophonic synthesizer?

If you recall last month's 'Synth Secrets', you'll remember that there are two types of 'monosynth' triggering: single and multi. The Odyssey is a multi-triggering synthesizer, so it generates a trigger every time you play a note, even if another note is (or notes are) already depressed. Armed with this knowledge, let's return to the Odyssey itself and repeat the experiment shown in Figure 4.

We'll start by defining amplitude and filter contours for the sound we're playing. For now, let's assume that both are the same, or the following explanation becomes too complex. The contour is a simple Attack, Decay, Sustain, Release (A=5, D=5, S=5, R=5), and it looks like Figure 5.

Now, if you played the sequence in Figure 4 on a truly polyphonic synthesizer, you would expect each note to be shaped individually, with each one following the ADSR contour shown. Unfortunately, on the Odyssey nothing could be further from the truth. The horrifying reality is shown in Figure 6.

Ghastly, isn't it? Both sets of notes (the upper four, and the single drone underneath) start healthily enough but, at the end of the first upper note, the apparent 'upper' amplitude drops instantaneously to zero. At the same moment, the apparent amplitude of the drone jumps up. This is because both oscillators are now contributing to the lower note. The situation remains like this until you play the second upper note, at which point the upper amplitude leaps back to the sustain level, the lower note drops to the sustain level, and both go into a truncated attack phase from the sustain level. This repeats for the third and fourth upper notes until, finally, the combined sound dies away when you release the drone. As I said, it's horrifying.

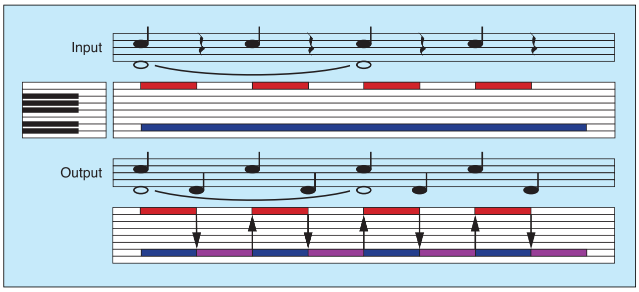


Figure 4: Playing a duophonic synth monophonically.

Yet... it has its uses. Imagine that you're trying to play a line that's so fast and convoluted that on a conventional monosynth you simply can't release the notes quickly enough to let the next articulate correctly. On a lowest-note priority monosynth like the Minimoog this would be a bit of a disaster, but the Odyssey separates its oscillators and re-triggers the next note whether or not you've managed to lift your fingers off the previous one. It then recombines the oscillators as soon as your fingers catch up. This means that, as long as you play the note-*ons* at the right moments, you can afford to be less concerned about the note-*offs*. What's more, if the overlaps between notes are not too long, it creates an interesting musical effect as one note merges into the next. Problem solved.

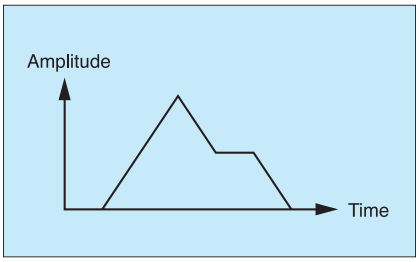


Figure 5: A simple ADSR contour.

## Greater Sophistication

It wasn't long before other manufacturers recognised the potential of duophonic architecture. Among the more esoteric models, there soon appeared the EMS VCS3 with its duophonic DK2 keyboard, the Octave CAT and its successors, the OSCar, and even ARP's duophonic 3620 keyboard for the ARP2600.

Perhaps the most notorious of these was the CAT, which bore striking similarities to the Odyssey. After threats of legal action from ARP, Octave replaced the original CAT with the SRM (Series Revision Model) and SRMII, which offered yet another form of duophony that Octave called 'two-note memory'. This grandiose name referred to a new keyboard scanning circuit that, if you played a two-note chord, memorised the interval between them and continued to apply it even if you then returned to playing monophonically. A form of programmable detune,

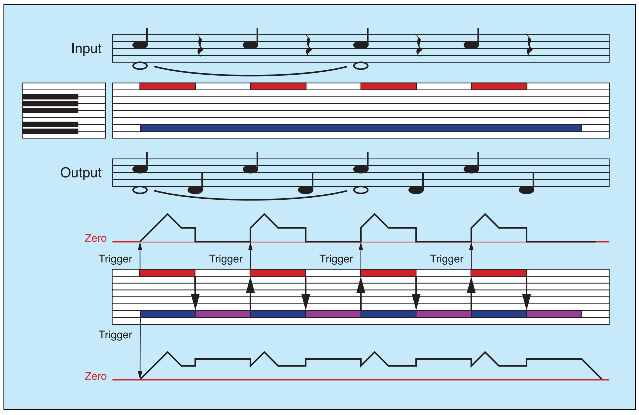


Figure 6: Shaping the notes (or not) on an ARP Odyssey.

this allowed you to switch between, for example, tunings in fourths and tunings in fifths, without having to adjust the oscillators themselves.

But it was the Japanese who took duophony to its next level. The Korg 800DV, the Roland SH7 and the Yamaha CS40 were not just duophonic, but duo-timbral. Of these, the first to appear, yet in many ways the most sophisticated, was the Korg. Released in 1975, the 800DV didn't copy ARP's primitive duophonic architecture — it proved to be much, much more.

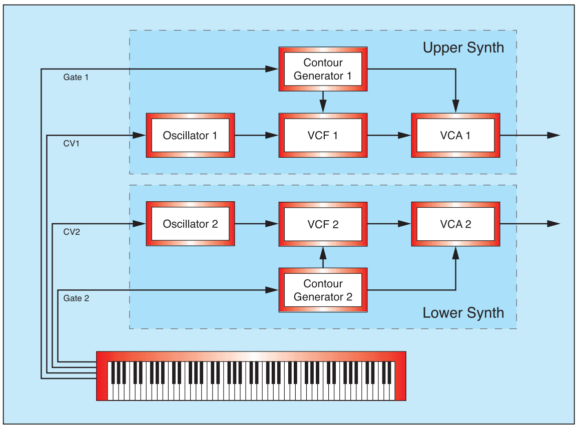


Figure 7: The basic duo-timbral structure.

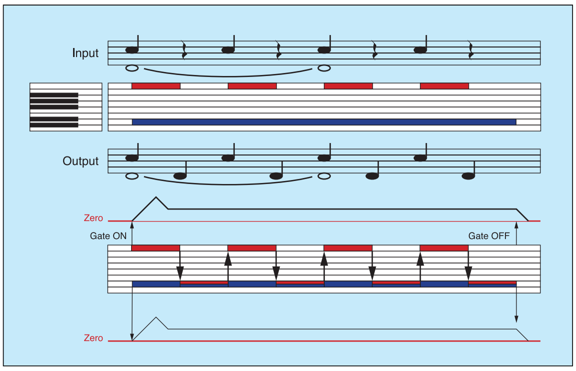


Figure 8: The duo-timbrality of the 800DV.

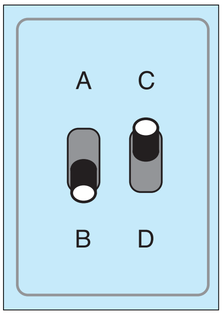


Figure 9: The 800DV's Key Transpose panel in BC mode.

## Duo-Timbrality

To understand the difference between the Odyssey and the 800DV, you need only compare Figure 7 to Figure 3. Whereas the Odyssey had a single signal path, the 800DV was two distinct synthesizers living inside a single box.

Although Figure 7 is greatly simplified (like the Odyssey before it, the DV800 incorporated LFOs, ring modulators, and so on) it demonstrates an important principle: that of duo-timbrality. Quite distinct from duophony (which means two notes), duo-timbrality means that you can create two independent sounds, each with its own oscillators, filter settings, amplifier settings, and contours. What's more, you can play each of these independently from the keyboard.

Or can you? Surely, when you release one of the two notes you're playing on an 800DV, the single remaining note will again become both the highest and the lowest, and we'll be back to square one?

To understand why the 800DV is (in this area) so superior to the Odyssey, let's first look at the triggering it employs. Or rather, let's not. That's because the 800DV has no triggers — only gates that remain 'high' for the entire duration of any continuously or contiguously held notes. This gives a very different result if you play the sequence shown in Figures 4 and 6. (See Figure 8.)

What happens in Figure 8 is distinct from Figure 6 in two ways. Firstly, there are no triggers to re-initialise the contour generators. Consequently, after the attack and decay phases, and while the notes continue to be held, the gate maintains the contours at their sustain levels. This eliminates the unnatural 're-firing' of the lower drone throughout the sequence. Secondly, because the oscillators lie within distinct signal paths, you don't get the unnatural combination of both when you play a single note. Although both signal paths produce the D when you release the upper C, the distinct tonalities of the two patches survive.

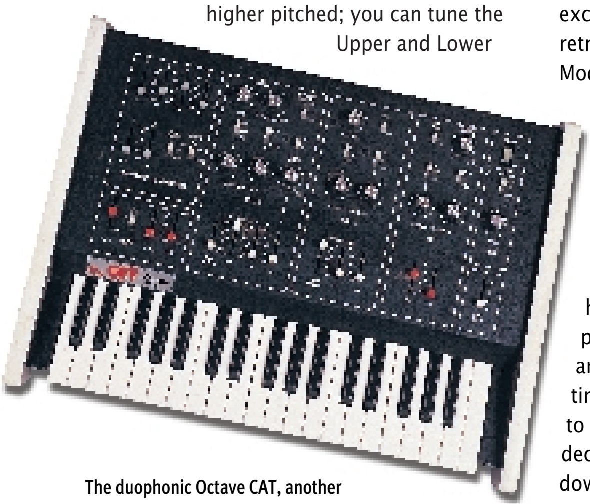
If this were the only improvement over the Odyssey, the Korg would still be more usable as a duophonic instrument.

But there's far more to the 800DV than this, some of which is embodied in the Key Transpose section on the far right of its control panel.

Key Transpose (see Figure 9) is a badly named facility from the heyday of Japlish manuals, and it hides some unique facilities. Let's start by putting the two switches in the AC position. In this mode, pressing just one key causes both of the 800DV's internal synthesizers to sound together, whereas pressing two keys will split the timbres into Upper and Lower notes. Notwithstanding the separate signal paths, this is the same as the Odyssey.

AD mode is something new. In this mode, only the Lower synthesizer sounds when you press one key, and both are produced when you play two. This is a huge improvement, because it allows you to play two parts rather than just two notes. Now we're getting close to true two-note polyphony. Sure, there are limitations, and the two notes are not correctly articulated in all circumstances. In particular, the tails of the Upper synthesizer can drop down to the Lower pitch when you release the Upper, but if you make Release = 0 the output from the 800DV begins to look much like the input sequence. (See Figure 10.)

BC mode is identical to AD mode, except that it is the Upper synthesizer you hear when just one key is pressed, and the Lower joins in when you press a second. Not much use for our example sequence, but ideal for the times when you want to hold a drone with your right hand and play a moving line with your left. (This does not necessarily mean that the right-hand note is the higher pitched; you can tune the Upper and Lower synthesizers to any octaves and pitches you choose.)



The duophonic Octave CAT, another early 2-note synthesizer.

Finally, we come to BD mode. It works like this: play one note and nothing happens; play two and you hear both. (See Figure 11.)

## The Repeat Function

You might think that we have now plumbed the depths of duophony — but you would be wrong. If you look underneath the Key Transpose section on the DV's control panel, you'll find another section, called Repeat. This offers two sliders, plus a Mode selector knob with six positions. (See Figure 12.)

The action of the left-hand slider is obvious: it's a clock. The action of the second slider is less obvious, but will become clear in a moment. As for the Mode selector... well, let's look at each of the modes in turn.

Mode A leaves the Lower voice unaffected, but retriggers the Upper voice at the clock rate, with the length and position of the Gate defined by the second slider. (This is important because the start of the 'next' cycle truncates the current Gate, so you can alter the character of the sound by moving the second slider.)

Mode B is the same, except that it is the Lower voice that is retriggered, while the Upper is left unaffected. Mode C is similar, but retriggers both voices simultaneously. Things get even more interesting with Mode D, because this retriggers the voices alternately. (See Figure 13.)

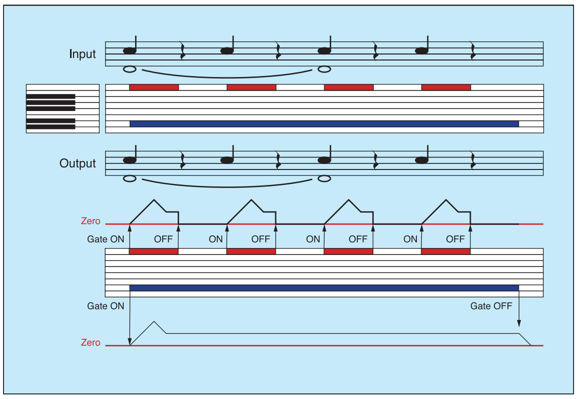


Figure 10: The Korg 800DV in AD mode.

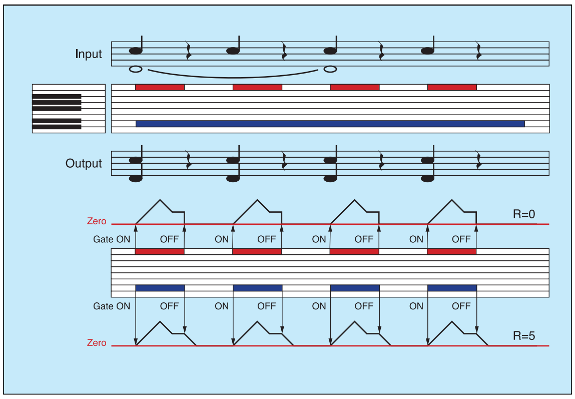


Figure 11: BD mode only opens a Gate to play the notes when two keys are depressed.

## A Practical Application

Before looking at mode E, let's imagine that you have programmed the Upper synthesizer to produce a high-pass filtered noise burst whose amplitude decays rapidly to silence. At the same time, you've programmed the Lower synthesizer to produce a tonal sound that decays more slowly, and swoops downward in pitch as it does so. Put the two together and, if you set everything correctly, you have the components of many classic analogue drum sounds.

Play one note in the Key Transpose AC mode and you will hear both sounds together — an analogue snare. One note in BC mode gives the Upper voice only (which you can tune to sound like an analogue hi-hat), while one note in AD mode gives the Lower voice. This, played at the bottom of the keyboard, can sound like an analogue kick drum. Played across the middle register, it imitates

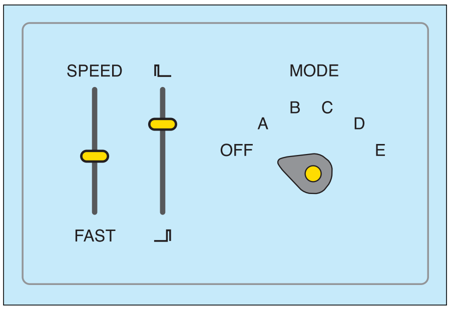


Figure 12: The Repeat panel from the Korg 800DV.

analogue toms.

Next, let's add the Repeat Mode. There are 24 permutations of Key Transpose and Repeat, so we'll consider just one of these: AD and C. The Key Transpose mode determines that the Lower sound (the 'body' of the drum) will sound whenever you press a key, and that the Upper sound (the snare or hi-hat) only sounds when you press a second key simultaneously. At the same time, the Repeat mode ensures that both the Lower and Upper repeat whenever you hold a key for a reasonable length of time.

Putting the two together you'll find that, if you hold one key, Lower repeats with a steady kick beat, but there is no sound from Upper. If you then play rhythmically on a second, higher, key, you can produce hi-hat-type patterns. If you hold Upper for an extended period, it too repeats on the beat. Of course, you can mix and match these responses to create new patterns, and experimentation with the other modes will yield different results.

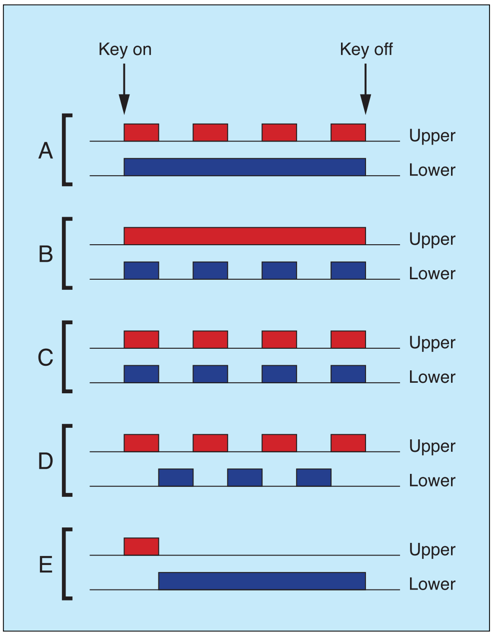


Figure 13: Korg DV800 Repeat patterns.

## From Duophonic To Digital

Finally, let's consider Repeat mode E. From Figure 13 you can see that this plays a single, curtailed instance of the Upper voice, followed by the Lower voice.

Let's imagine that you want to program a composite sound with two quite different tonal characteristics. You can't do this on a monophonic synthesizer because, flexible as filters, amplifiers, and ADSR envelope generators may be, they don't permit you to change the fundamental nature of a sound in mid-note. However, mode E allows you to do precisely this, with the rate at which the Lower synthesizer replaces the Upper one determined by the repeat speed and the position of the gate width control.

Although we didn't realise it at the time, the 800DV was the first instrument to introduce the concept of 'partials'. For example, you could use Upper to create a noise-based snippet of sound — a 'chiff' — at the start of a note, followed quickly by a tonal sound produced by the Lower synthesizer. This is exactly the same concept that Roland re-introduced more than a decade later when they released their first digital synthesizer, the D50. The D50 used a PCM sample for the attack partial and a digitally generated waveform for the sustain, but the principle was the same.

Consequently, although I stated above that "duophonic synths are far more limited and far less intuitive than you might expect", the duo-timbral Korg 800DV demonstrates that I must modify this month's Synth Secret considerably. It now reads:

*Duophonic synthesizers are far less than polyphonic synthesizers restricted to playing two notes at a time, but duo-timbral synthesizers are far more than monophonic synthesizers that can play two notes at a time.*

Logically enough, next month I'll be following up the last two issues' discussions of monophony and duophony with an article about polyphony! SOS

# synth secrets

## PART 20: INTRODUCING POLYPHONY

Having explored the way monophonic and duophonic analogue keyboards work, **Gordon Reid** puts away his Minimoog and Odyssey and descends into the complex world of polyphonic synths to a flourish of complex jazz chords.

"Just like human beings, sounds are born, reach their prime, and die; but the life of a sound, from creation to evanescence, lasts only a few seconds."
*...taken from my Yamaha GX1 manual, 1975.*

Ah, polyphony. You wouldn't think that there's much to it, would you? After all, we were all brought up with out-of-tune Edwardian pianos, Bontempi organs, five-string acoustic guitars, and whatever else lurked unloved behind the sofa in the living room. You hit a note... it went 'boiinnggg'. You hit another note... it too went 'boiinnggg'. It must be the same on a synthesizer, yes? Well, no, otherwise I wouldn't have asked.

Before we can analyse and judge the various ways in which synthesizers have achieved polyphony, we had better understand precisely what 'polyphony' is. So I'll let you into a secret right away... just because an electronic instrument can play many pitches simultaneously, it isn't necessarily truly polyphonic.

### Uhh...?

Let's remind ourselves of what happens when, for example, you hit a strike a string within grandma's piano or pluck a string on an acoustic guitar. As we learned in the very first part of this series, the sound thus produced will have a characteristic tone determined by the nature of the string, and it will vibrate at a particular set of pitches determined by its length and tension. Of course, this isn't the end of the story. Percussive instruments such as these are loudest at the start of a note and, unless damped, their sounds die away to silence over the course of several seconds. Furthermore, the note is brighter (ie. contains more high-frequency harmonics) at its start than it is at its end. In addition to this, many such sounds fluctuate in some way, exhibiting modulations such as vibrato.

Numerous other factors determine the exact sound produced, so it's both surprising and comforting to know that we can reduce these factors down to three major attributes for many simpler timbres. The first of these is the principal waveform, which provides both the initial tone and the pitch. The second and third are the changes in brightness and loudness as the note progresses. Given this, we can design a simple, monophonic synthesizer, as shown in Figure 1 above.

This has an oscillator that creates the basic waveform, a filter that controls the tone, and an amplifier that controls the loudness. A contour generator determines how the filter and amplifier modify the sound as it develops over time, and the modulator adds vibrato, tremolo and/or growl. It's a very elegant design and — if the oscillator offers at least two or three initial waveforms — it will produce a huge range of imitative and 'electronic' sounds. Correctly set up, it may even provide passable imitations of our hammered and plucked strings.

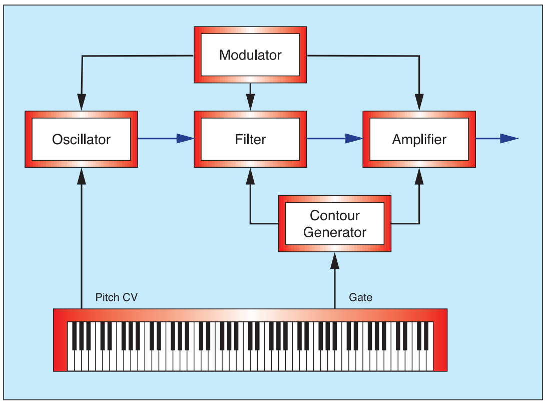


Figure 1: A simple monosynth.

Of course, by virtue of its single oscillator, the synthesizer in Figure 1 is capable only of producing one pitch at any given moment. Therefore, as I showed in part 18 of this series, back in October, its response to multiple notes is to play just one at a time, as determined by the key priority (see Figure 2, right). Clearly, no matter how many keys you press simultaneously, this can never be a polyphonic instrument.

Let's now return to our initial consideration of a piano or acoustic guitar. Imagine that you play a

Moog's Polymoog was a truly polyphonic analogue synth, but it was fantastically expensive and beset with reliability problems, for reasons which will be clear if you read on this month. See *SOS* June 1998 for the full Retrozone article, or surf to: www.sospubs.co.uk/sos/jun98/articles/polymoog.html

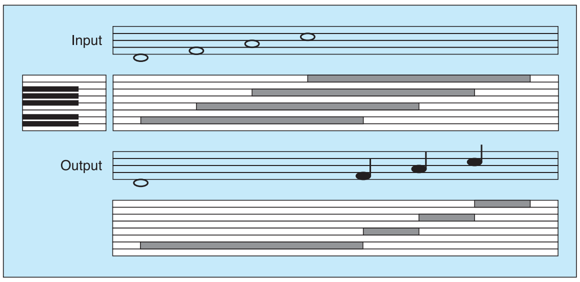


Figure 2: A four-note broken chord played on a low-note-priority monosynth.

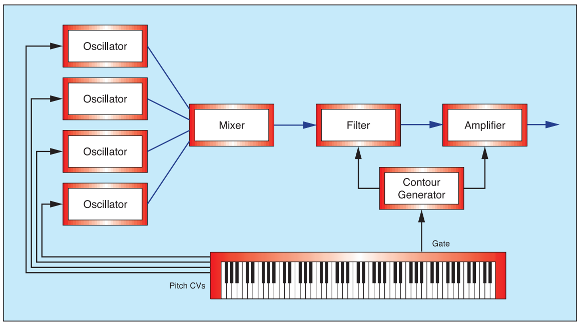


Figure 3: A four-oscillator synthesizer.

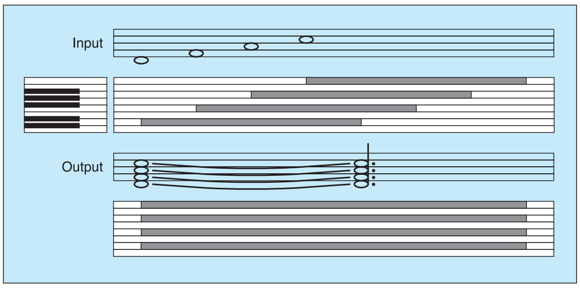


Figure 4: Why Figure 3 isn't a polyphonic synthesizer.

single note — say, middle C — and listen to the way that it develops over time. However, before it dies away completely, you press/pluck another note — say, the G above middle C. Initiating the second note does not affect the first... it's a complete entity in its own right. So let's add more oscillators and more pitch CVs to the design in Figure 1. Surely we've then done everything necessary to let us produce multiple instances of our imitation of the stretched string? In other words, we have designed a polyphonic synthesizer, haven't we?

Unfortunately, no. This design (see Figure 3) has numerous flaws. To understand the most important of these, remember that an analogue synthesizer's oscillators are *always* oscillating. On any commercial instrument, pressing a key on the keyboard may determine the pitch at which they do so, but it does not 'switch them on and off'. The amplifier at the end of the signal chain does this. Therefore, if the amplifier in Figure 3 is passing the sound of any one oscillator, it is passing the sound of all the others.

Figure 4 shows the result of this, and the result is the same, whether you're pressing one key, two, three... or a hundred. The multiple pitch CVs in Figure 3 may enable you to control the pitches of the oscillators, but you can't articulate them independently. It doesn't matter what contour you are using, or how you set the filter; this design is not a suitable basis for a polyphonic synth.

Let's recap... To imitate a moderately complex monophonic sound, or even create a new 'electronic' timbre, you need a minimum complement of an oscillator, a filter, an amplifier, a contour generator and a modulator. And, from the argument leading to Figure 4, it's clear that the presence of multiple pitch generators (ie. lots of oscillators) is insufficient to create true polyphonic synthesis.

As you might imagine, there are several ways to overcome this. But, before we do the obvious thing and throw multiple oscillators, filters, and amplifiers at the problem, we're going to step backwards and discuss an electronic instrument that isn't a synthesizer at all. It's the lowly electric organ.

## The Organ-ic Approach

The neat thing about the electric organ is that, unlike most synthesizers, every key has the equivalent of a dedicated oscillator. The most common design uses 12 master oscillators that output high-frequency signals related to each of the keys C, C#, D... and so on up to B. The outputs from each of the master oscillators then pass ▶

▶ through devices called octave dividers. These reduce the initial frequencies by factors of two, four, eight... and so on, thus creating the correct frequencies for all of the Cs across the keyboard, all of the C#s, all of the Ds... and so on (see Figure 5, right).

Since each key on the organ's keyboard has a dedicated sound source, we can treat each key as an on/off switch that allows a desired pitch to enter the signal chain (or not). Indeed, there's no reason why the contacts underneath each key should not be exactly that... switches in the signal path itself. We can therefore design a keying system, as shown in Figure 6. This is in marked contrast to the switches underneath a synth keyboard, which provide pitch CVs, Gates, and Trigger pulses, but have no direct contact with the audio signal.

Consider how this works. You press a key — say, C1 — and a circuit is completed. This allows the appropriately divided waveform generated by the master C oscillator to pass to the mixer. The signal then passes to the amplifier, and thence to the outside world, whereupon you hear it go 'beeeeep'.

Now, while still holding the C, you press another key — a G. This time, the appropriately divided waveform from the master G oscillator passes to the mixer, and onwards to the amplifier. You hear a two-note chord. Now press an E. You hear this note added to the chord. Now rest a copy of War And Peace on the keyboard. You hear a cacophony of notes... you get the idea.

To continue the experiment, let's release the original C1. No problem... you lift your finger, and the note stops without affecting any of the others that are still playing. Clearly, this organ is truly polyphonic in the sense that (i) you can play simultaneously as many notes as you wish, and (ii) every note is independent of all the others. This is because — in effect — every note has an associated amplifier with two states: Gain=1 (you press a key and complete the circuit) and Gain=0 (you release the key and *break* the circuit). So let's redraw Figure 6 using amplifiers and Gate signals (see Figure 7).

## Organs Are Boring, Synths Are Not

Unfortunately, the notes produced using our simple electric organ are not very interesting. For one thing, they have a constant tone, and the VCA contour is a simple on/off shape (as shown at the top of Figure 7). To put it bluntly... it's a cheap, nasty organ, just as we designed it to be. However, it still has that one, huge advantage over all the synth designs we have discussed: it is truly polyphonic.

So let's add back the filter, amplifier, contour generator and modulator from Figure 1 (see Figure 8). Surely, we now have a polyphonic synthesizer? Umm... no (again). Figure 8 represents a form of sound generation called 'Paraphonic' synthesis, prevalent in the late '70s and early '80s. Perhaps a better name would be 'Quasi-Polyphonic'; but that hardly trips off the tongue, so 'paraphonic' it is.

But why isn't 'paraphony' (if there is such a word) the same as polyphony? The answer to this is obvious if we consider the articulation of individual notes played on the instrument in Figure 8. Look at the components to the right of the mixer. As always, these shape the notes produced by the sound generating part of the electronics — ie. all the stuff to the left of the mixer. Let's consider a note played in isolation:

- You press a key.
- This opens the appropriate VCA (ie. sets the Gain to 1) and passes the appropriate pitch to the mixer.
- At the same time, the second Gate (or Trigger) initiates the ADSR contour generator.
- The contour generator begins its Attack stage, followed by its Decay, and settles at its Sustain Level.
- The output from the mixer is shaped in both tone and loudness by the A, D and S stages, and you hear the note develop as it does so.
- You release the key.
- The note stops immediately, because the VCA Gain before the mixer immediately drops to zero. This is irrespective of the Release value of the contour generator.

OK... the Release is a bit of a problem, but it's easily corrected ▶

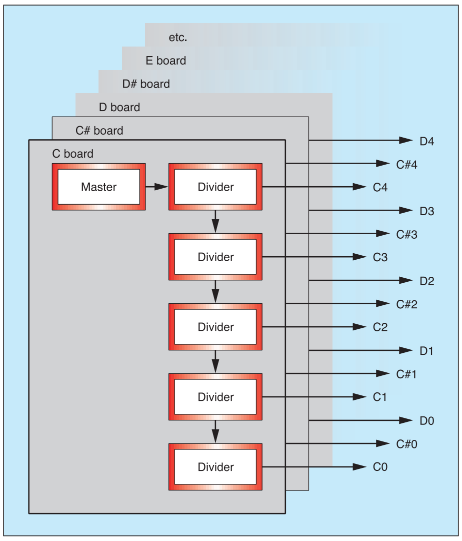


Figure 5: Generating the full range of pitches using octave dividers.

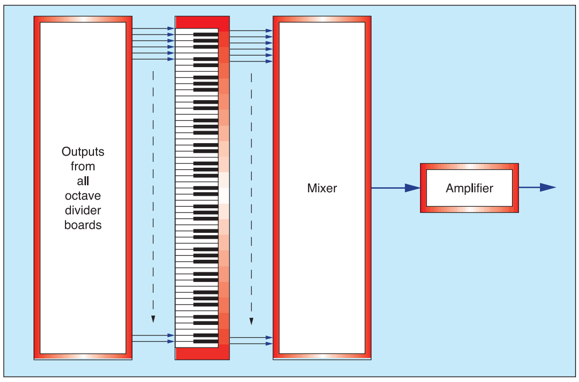


Figure 6: The simple 'divide-down' organ.

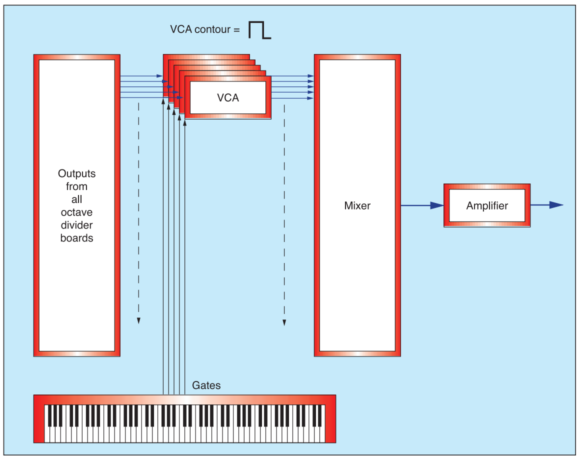


Figure 7: An organ that uses VCAs to let notes 'speak'.

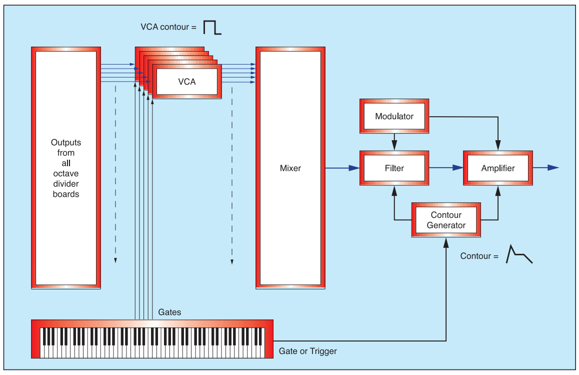


Figure 8: A 'Paraphonic' synthesizer.

by changing the release time of the pre-mixer VCA to be equivalent to the maximum release time of the post-mixer contour generator. We couldn't do this if we were using the keyboard-switch architecture of Figure 6, but there's no reason why we shouldn't do it in Figure 8. Indeed, we don't even need a dedicated contour generator to do this — just a VCA whose gain CV leaks away relatively slowly once the Gate has been released (it's highly unlikely that you would ever design a synth like this, but let's not allow facts to interfere with a good argument).

Having made this adjustment to the VCAs in Figure 8, let's return to the experiment, extend our thoughts to include two notes played together, and see what happens:

- You press the first key.
- As before, this opens the appropriate VCA and passes the appropriate pitch to the mixer.
- At the same time, the second Gate (or Trigger) initiates the ADSR contour generator.
- The contour generator begins its Attack stage, followed by its Decay, and settles at its Sustain Level.
- The output from the mixer is shaped in both tone and loudness by the A, D and S stages, and you hear the note develop through to its Sustain.
- Now you press a second key.
- Unlike the first, this does *not* follow the ADS contour stages, because there is only one contour generator, and it has already reached the Sustain level. Consequently, the brightness and loudness of the second note immediately settle at the Sustain level.
- You release the first key.
- The volume of the first key decays to zero at the rate defined by the initial VCAs.
- You release the second key.
- This and the tail of the previous note decay to zero at the rate defined by the contour generator in the secondary part of the synth.

This may sound very confusing when written as 11 bullet points, but Figure 9 (above) should clarify matters.

Clearly, paraphonic synths are not truly polyphonic. Sure, we're much closer than before, and the output score in Figure 9 looks like the input score, but the 'shapes' of the individual notes (ie. the way their perceived loudness and brightness develops over time) are still wrong. Returning to our thoughts at the very beginning of this article, imagine the surprise you would feel if, when you played multiple notes on a piano or a guitar, the second and subsequent ones lacked their natural brightness and loudness profiles. It couldn't happen.

You might therefore think that the architecture in Figure 9 is a silly way to build a synth, but there are many instruments very similar to this. The one that coined the name 'paraphonic' synthesis appeared in 1979. This was the Roland Paraphonic RS505. Others include the earlier Roland RS202, the Korg Polyphonic Ensembles, the ARP Omnis, and even the revered Solina String Ensemble.

## String Synthesizers

Ah yes... string ensembles and string synths. It's no secret that I loved (and still love) these simple keyboards, partly because — at a time when a Mellotron was an unaffordable dream — there were examples of the genre that did not suffer from the 'paraphonic' limitation. Of these, the best (in my opinion) was the Logan String Melody II, a gorgeous instrument that shaped the 'loudness' of every note correctly, no matter how many you played simultaneously (see Figure 10).

The Logan did this by providing a dedicated AR contour generator for the VCAs controlling each note, thus shaping each one individually. Furthermore, the Attack and Release rates were under the player's control, so you could make notes as 'square' and organ-like or as slow and languorous as you desired. The String Melody II made full use of this flexibility by providing a selection of string and organ tones that you could contour as required.

On paper, it looks like a good system, and it is. Logans and other string machines that use this system (such as the Hohner String Performer and the Godwin String Concert) are, in my opinion, vastly superior to more famous paraphonic designs such as the Solina, the Rolands, and the Omnis.

Unfortunately, there's a problem. Although the Logan's architecture can

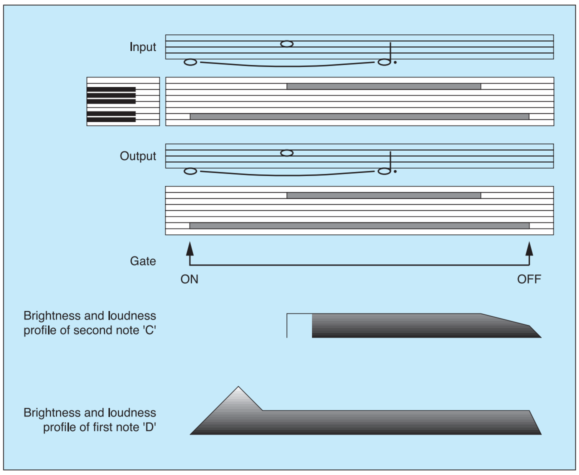

Input
Output
Gate
ON
OFF
Brightness and loudness profile of second note 'C'
Brightness and loudness profile of first note 'D'

Figure 9: Profiles of notes played on 'paraphonic' synths.

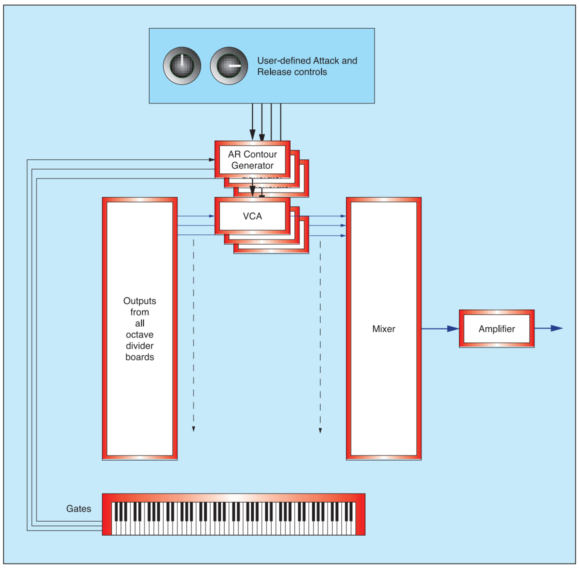

User-defined Attack and Release controls
AR Contour Generator
VCA
Outputs from all octave divider boards
Mixer
Amplifier
Gates

Figure 10: A truly polyphonic 'string synth'.

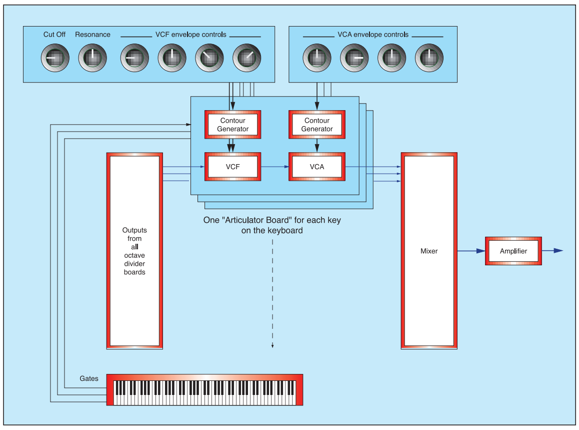


Figure 11: A true polysynth.

articulate the *loudness* of each note correctly, there's still no way to make it articulate the individual *tone* (or brightness) of the notes. To do this, we must add a dedicated VCF and a second Contour Generator for each note, as shown in Figure 11 on page 84.

At last, we're getting somewhere. Each time you play a note, the individual sound produced by the master oscillators and octave dividers is shaped by a dedicated filter and a dedicated amplifier, each controlled by its own contour generator. This means that each note is articulated independently of all the others, and the nastiness that is Figure 9 now becomes the true polyphony of Figure 12 (right).

So, taking my cue from the GX1 manual, here's this month's Synth Secret:

*Just like human beings, sounds are born, reach their prime, and die. Furthermore, if you have more than one person in a room, each must be born, reach his/her prime, and die independently of all the others.*

It's obvious, really.

## Real Synthesizers

Let's finish this month's Synth Secrets by taking a brief look at two synthesizers that adopted this approach to polyphony. The first of these is the grandiosely titled Polymoog Synthesizer, a much maligned instrument that doesn't deserve the opprobrium that has been heaped upon it. Ignoring subjective views of its strengths (or lack thereof) the Polymoog has a unique architecture based on Figure 11. However, instead of providing user controls for the filters and amplifiers in each of the 'articulator' boards, it offers a handful of presets. This means that, when you press one of the buttons '1' to '8' in the centre of its control panel, you send predetermined CVs to the boards. Despite this minor limitation, the articulators still shape every note correctly. The Polymoog is a truly polyphonic synthesizer.

*However*... (you just *knew* there'd be one of those, didn't you?), to this excellent architecture, Moog added a single VCF with a dedicated contour generator, another VCA with a dedicated contour generator, and a set of EQs called Resonators (see Figure 13). You might think that this would make the instrument even more powerful but, if you study the diagram, you can see that the designers did something really weird — they embedded a true polysynth within a paraphonic synthesizer! It's small wonder that many players believed (and still believe) that the Polymoog was not truly polyphonic. If they had stuck to using the 'Mode' output and left all the controls in their 'Pre' states, they would have been able to play it in true polyphonic fashion. Unfortunately, almost everybody used the final VCA, the VCF and the Resonators, thus making the Polymoog 'merely' paraphonic. It was doomed.

The second synth in my tale is as revered as the Polymoog is derided. It is the Korg PS3200, the middle sibling in a family that comprises the PS3100, PS3200 and the mighty PS3300. The 3200 combined 'total' polyphony (you could press every note on its 48-key keyboard simultaneously, and each would be articulated correctly) with modularity (you could patch large chunks of it), and included 16 patch memories, too. Of course, the memories couldn't recreate any missing patch cables, but no matter... the PS3200 was, and remains, a unique, fantastic, dreamy... well, you get the picture. If you are an analogue *aficionado* and you see a good one of these at a sensible price, buy it.

The important point here is that the PS3200 had all of its filters and amplifiers *before* its mixer, thus retaining true polyphony in all circumstances. Indeed, its architecture (notwithstanding a bucketful of extra facilities) is so similar to Figure 11 that we need no extra diagram to describe it.

## Epilogue

So there we have it... the secret of polyphonic synthesis. However, having ploughed through all that theory, you may be dismayed to learn that it was seldom put into practice — very few synthesizers adopted this approach. There are two reasons for this. Firstly, there's the cost. Imagine it... under every key — figuratively, if not geometrically — there's a dedicated 'articulator' comprising at least one filter, at least one amplifier, and at least two contour generators. Even with the advent of ICs, this architecture is very expensive. The second reason is just as important. It's reliability. If you have a synth designed with 48 or even 72 individual 'articulator' boards, you have many, many times the chance of a breakdown compared to simpler instruments.

Indeed, I can't think of any synths — other than the Polymoogs and the Korg PS3000-series — that used this design. All the other analogue polysynths — Prophets, Oberheims, Yamaha CSs, Roland Jupiter Xs, the Memorymoog — all used a different method. Indeed, as I write this, I can think of three other approaches, all similar, but all different. We'll discuss those next month. SOS

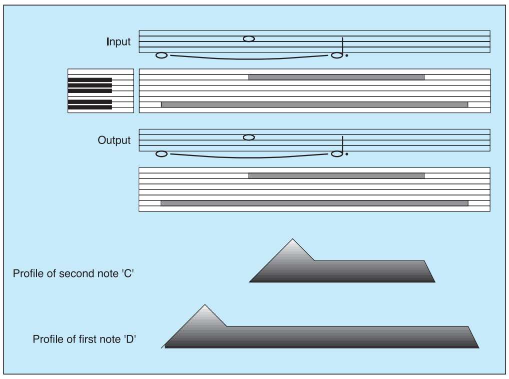
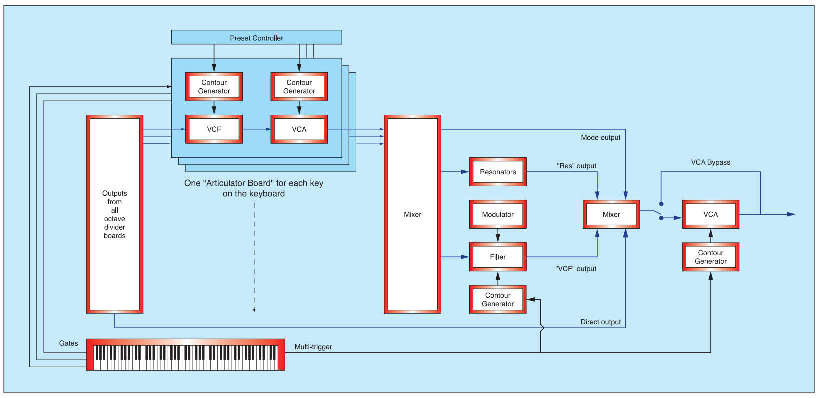


Figure 12: Profiles of notes played on a truly polyphonic synth.

Figure 13: A simplified representation of the Moog Polymoog.



# synth secrets

## PART 21: FROM POLYPHONY TO DIGITAL SYNTHS

Polyphony is hard to achieve on analogue synths without incurring hideous expense. This month, **Gordon Reid** explains how synth manufacturers employed digital technology to overcome this problem.

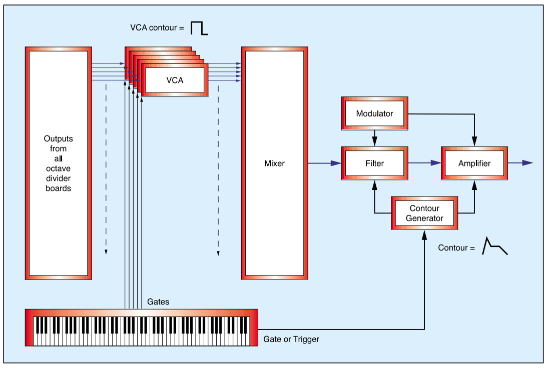


Figure 1: A paraphonic synthesizer.

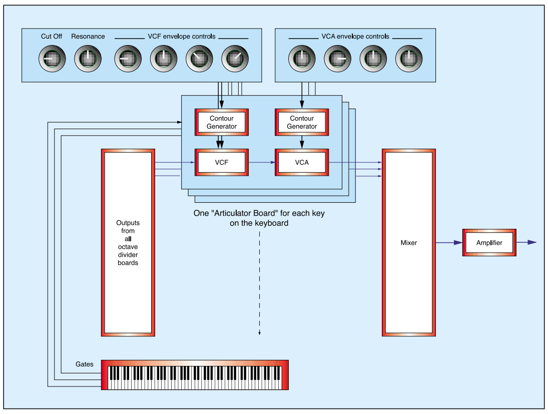


Figure 2: A fully polyphonic analogue synthesizer.

Last month, I showed that a truly polyphonic synth must be able to shape the tone and loudness of any sound independently of any other sounds or notes that it is producing at the same time. We then looked at one way in which an analogue synth can achieve this — ie. by providing filters, amplifiers and contour generators for every key on the keyboard. Unfortunately, an instrument of this design is expensive to manufacture, and its complex architecture usually results in unreliability. Moog provided total polyphony in this way on their Polymoog, and sure enough, this keyboard suffered from being too expensive and unreliable.

Unsurprisingly, most other manufacturers adopted different ways of circumnavigating this problem — and this month, I'll try and explain how exactly they did this. However, before considering the most common polyphonic architecture, I would like to look at one of the strangest analogue keyboards ever devised. It's not particularly obscure, nor very rare, but it has the weirdest architecture I've ever encountered. It's the Crumar Trilogy.

### Crumar's Cut-down Paraphonic Approach

To understand the Trilogy fully, you'll need a quick reminder of the content of last month's Synth Secrets. Figure 1 shows the architecture of a 'divide-down' paraphonic synth on which only the first note played benefits fully from the Attack and Decay stages of the contour generator, and only the last note benefits from the Release. Figure 2 depicts a fully polyphonic 'divide-down' synthesizer — such as the Polymoog — that offers a VCF/VCA/EG 'articulator' board for every note on the keyboard.

As I've just recapped, the latter is expensive and potentially unreliable. A valid question is therefore: is it possible to devise an analogue synth that combines full polyphony and correct articulation of all notes, but with lower cost and greater reliability? Unfortunately, the answer to this is no, although the Crumar provides a unique and interesting compromise.

Let's go all the way back to octave-divider boards (see Figure 3, page 156). You'll remember from last month that each board uses a master oscillator related to a given note on the keyboard, and then 'divides down' that frequency to generate all the instances of that note across the keyboard.

Now let's take just one of these boards (the one that creates all the Cs, for example) and give it its own contour generator, filter and amplifier. The result is, in essence, a dedicated paraphonic architecture for all the Cs alone, and it looks like Figure 4 (see page 156).

If we now stack 12 of these boards — one for each note in the octave — we have a weird synth that articulates every note correctly, *provided* that you never play more than one instance of any given note. For example, if you play a C1/E1/G1 triad, each note will speak correctly whether you play them simultaneously, as a broken chord, as an arpeggio, or even within a solo. However, the moment that you add another 'C' to create the octave chord — C1/E1/G1/C2 — the second 'C' cannot be correctly articulated. This is because the appropriate contour generator, filter and amplifier have already received a Gate, and are some way along their ADSR response. I have shown this mythical synthesizer — the 'GR1' — in Figure 5 (see page 156).

The 'GR1' is mythical because, to the best of my knowledge, there has never been a commercial synthesizer that works in this way. The Crumar Trilogy, however, came close. If you ignore its organ and string ensemble sections (the basic sounds of which are tapped from the divide-down boards and passed through appropriate treatments to obtain the desired timbres), you find that its architecture looks very similar to the 'GR1'. However, there are two important differences.

Firstly, each set of notes has two independently tunable oscillators. The number of master oscillators and dividers is therefore double that shown in Figure 5 (see Figure 3, opposite).

The second difference is more radical. In what could only be a cost-cutting exercise, Crumar made the Trilogy not 12x paraphonic, but 6x paraphonic. This means that all the Cs and all the F#s share a contour generator, filter and amplifier, as do all the C#s and Gs, all the Ds and G#s... and so on up the octave (see Figure 6, below).

Unconventional though this architecture may be, it remains surprisingly usable. For example, on what other keyboard could you hold a drone — say, a low C — and play a solo above it which retriggers the drone every time that you play another C or an F#, but at no other time? Well... on no other. Add to that the Trilogy's ability to layer the strings and organ sounds derived from oscillator bank 1, and you have something quite unusual — not quite a true polysynth, but much more than a basic string machine or paraphonic synth.

Despite its curiosity value, the Trilogy possesses a structure which is clearly unsuitable for a fully polyphonic analogue synthesizer. Nevertheless, it demonstrates an important consideration in electronic design: if you use fewer components, your product will cost less and will be more reliable, all other things being equal.

## Oberheim & The SEM

So let's look at polyphony a different way — the way Oberheim chose to in the mid-'70s. Remember, for every note played on a fully polyphonic instrument, you need a dedicated sound source, dedicated filters, dedicated amplifiers, and dedicated contour generators. So how about cutting down on components, by reducing the amount of polyphony and then assigning a complete synthesizer 'voice' to each note as you play it? This seems like a good idea, but it raises problems of its own. In particular, if we're going to limit the polyphony to just a handful of notes... which handful do we choose? Of course, the answer to this has to be 'whichever ones you happen to be pressing at any given moment' — otherwise, the synth is unplayable!

The first affordable synth that allocated voices to notes was the Oberheim 4-Voice, and we're going to see how it did this. But before going any further, here's a health warning for analogue anoraks: for the first time in Synth Secrets, we're going to take a detour into some real digital electronics. This is unavoidable, because logic circuits are the only practical and economical way to determine which keys are depressed at any given moment on a polyphonic keyboard (this also explains why there were no polyphonic synthesizers before 1974... the necessary technology did not exist!).

## Binary Numbers & Keyboard Scanning

I'll start by refreshing your knowledge of binary numbers. As you know, binary is simply a number system, just like the decimal system used in everyday life. However, instead of counting from zero to nine before adding a new column, as in decimal, we only count from zero to one before adding a new column. Consequently, whereas the decimal system counts 0, 1, 2, 3, 4, and so on, binary represents the same numbers as 0, 1, 10, 11, 100... and so on. I have shown the equivalence of some decimal and binary numbers in Table 1, below.

| DECIMAL | BINARY |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |
| 16 | 10000 |
| 32 | 100000 |
| 64 | 1000000 |
| 128 | 10000000 |
| 255 | 11111111 |

As you can see, every time a number in binary is depicted using all 1s, we add another column of digits on the left and start again. This should be familiar in principle, as we do the same in decimal counting — when we reach 9, 99 or 999, we add a new column and start again at 10, 100 or 1000.

In digital electronics, people don't talk in terms of 'digits', but instead of 'bits'. Moreover, for any given number of bits we add all the leading zeroes to the binary number. Therefore, taking a decimal number from Table 1 — say, six — and writing it as an eight-bit binary value, we would write this not as 110, but as 00000110.

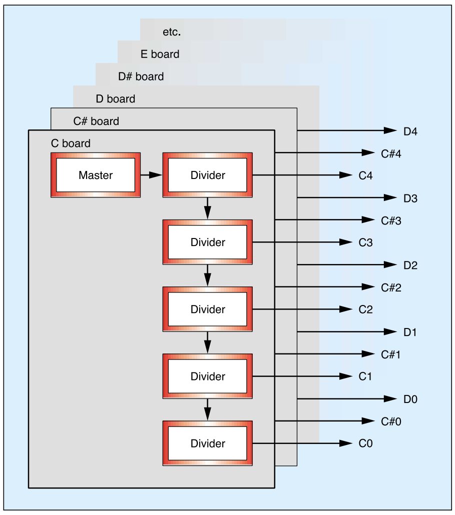


Figure 3: Octave divider boards.

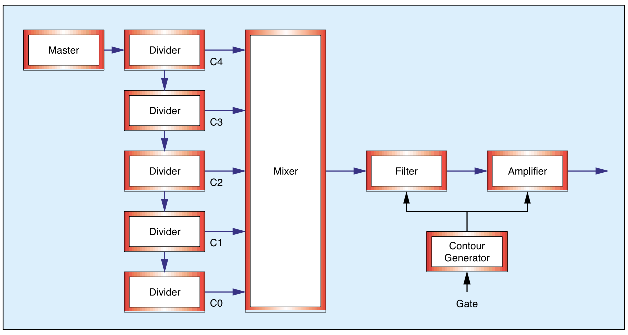


Figure 4: A paraphonic architecture for just the Cs on the keyboard.

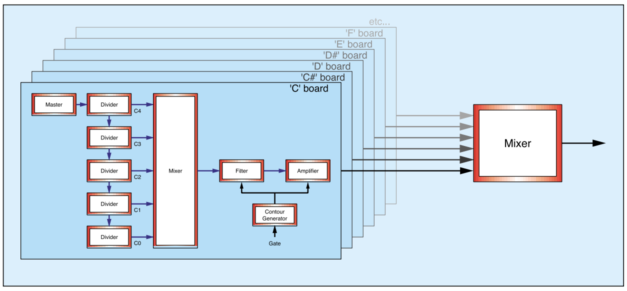


Figure 5: The mythical 'GR1' 12x paraphonic synthesizer.

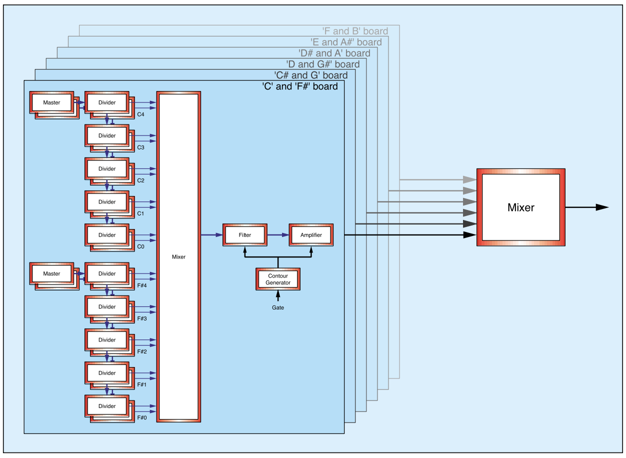


Figure 6: The synthesizer section of the Crumar Trilogy.

## The 6-Bit Keyboard Decoder

As we've discussed, binary numbers deal only with zeros and ones. But take a mental leap to the left, and you can just as easily say that binary numbers deal only with either/or states. These could be 'high' and 'low' voltages, or 'open' and 'closed' or 'on' and 'off' switch positions. And (ignoring for now such complications as velocity sensitive and pressure sensitivity) a keyboard is merely a rack of well-engineered on/off switches. So, here's another question: is it possible to devise a method that expresses the on/off status of a bunch of keys by representing them as a series of zeroes and ones?

The answer lies in Figure 7 (see right), which represents the 'digitally scanned polyphonic keyboard' developed in 1973 by Dave Rossum and Scott Wedge of Eμ (later renamed Emu) Systems.

If you've never studied digital electronics, you might think that you can't possibly understand this diagram, but I assure you that you can. Stick with me for a while longer, and I'll demonstrate how the 24 switches on the far right of Figure 7 can be used as the key contacts of a polyphonic synthesizer. Furthermore, I will perform magic, and show you that the single data line Z can carry all the information needed to tell the instrument which combination of keys is pressed at any given time.

But before moving on, we've got to discard all our previous understanding of monophonic and duophonic keyboard mechanisms. On most of these, the action of depressing a key closes a set of contacts that determine a pitch CV, as well as firing off a Gate and/or Trigger pulse. In contrast, a polyphonic keyboard scanning system must — by its very nature — relegate the keys to a passive role. The active element in the system is the digital (ie. binary) scanning circuit that recognises whether you have pressed any keys or not.

Now let's return to Figure 7. We'll start by considering what happens in Decoder X if you treat the three inputs D, E and F as a binary number. Let's assume that, if the voltage at the input is 'high' (+5V) it represents a 1 and if it is 'low' (0V) it represents a 0. Since there are three lines, we can use them to represent a 3-bit binary number. Table 2 below shows all the possible values.

| INPUT D | INPUT E | INPUT F | DECIMAL RESULT |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

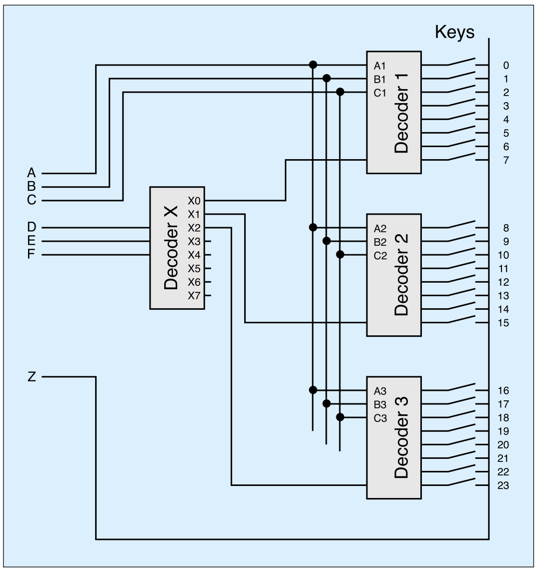


Figure 7: A simple 6-bit polyphonic keyboard decoder.

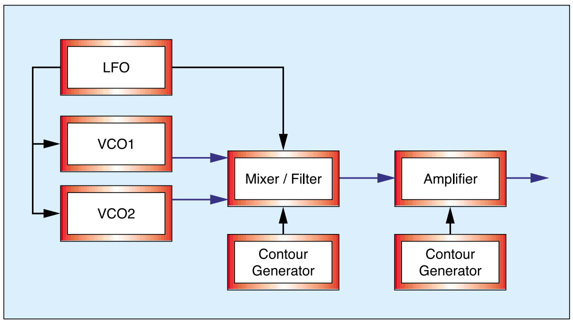


Figure 8: The basic modules within an Oberheim SEM.

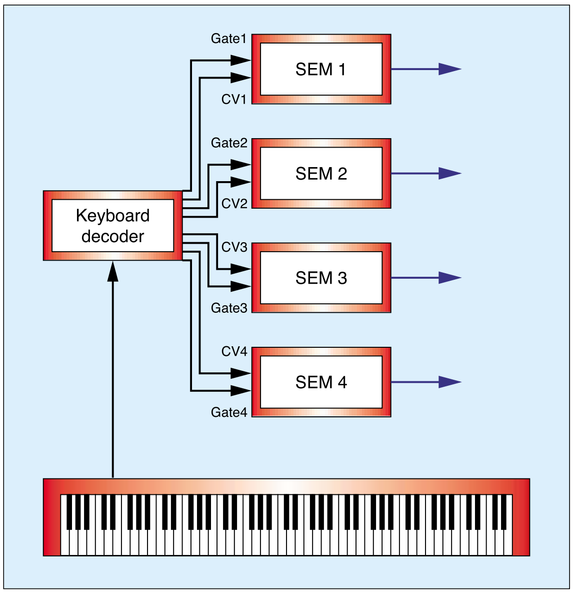


Figure 9: A 4-voice synthesizer?

Clearly, the three bits give a maximum of eight possibilities (including zero) so the three inputs D, E and F can determine which of (up to) eight outputs from Decoder X can be set 'high' at any given moment.

Now let's look at the lines A, B, and C. These are routed to each of the subsidiary decoders, Decoder 1, Decoder 2, and Decoder 3. Again, they can represent the eight numbers from zero to seven. Therefore, Decoder 1 sets its '0' output high when it receives '000' from A, B and C. It sets output '1' high when it receives '001' from A, B, and C... and so on.

The extra element is the line X0 that leads from Decoder X to Decoder 1. This is an 'enable' line, and it ensures that Decoder 1 will only do its stuff when X0 is 'high'. Likewise, Decoder 2 is only active when X1 is 'high', and so on.

Still with me? If so, let's now analyse what happens as we count in binary across the six data lines, D, E, F, A, B, and C:

- With DEF = 000, Decoder 1 is enabled, and A, B and C count from 0 to 7, in turn setting the voltage 'high' on the input to each of the switches 0 to 7.
- When the count reaches 000111, it continues from 001000.
- With DEF = 001, Decoder 2 is enabled, and A, B and C count from 0 to 7, in turn setting the voltage 'high' on the input to each of the switches 8 to 15.
- When the count reaches 001111, it continues from 010000.
- With DEF = 010, Decoder 3 is enabled, and A, B and C count from 0 to 7, in turn setting the voltage 'high' on the input to each of the switches 16 to 23.
- When the count reaches 010111, the system resets to 000000 and the cycle begins anew. Mind you, if I had enough paper, I could count up to 111111, use eight subsidiary decoders, and have a maximum of 64 switches!

Now for the final piece of the jigsaw. Let's say that, for example, the input data DEFABC is

001001. This means that Decoder X enables Decoder 2, and that key 9 is 'high'. The important thing here is that, across the whole keyboard (yes... those switches in Figure 7 represent real keys on a synth keyboard) only key 9 is 'high'. But if you *don't* have key 9 depressed at that moment, the switch is open, and the voltage on the Z line (the output) remains 'low'. If, on the other hand, the key is depressed, the voltage on Z is 'high', and the system detects the status of the pressed key.

This result is beautifully simple and elegant. Simply by counting in binary from 000000 to 111111 we can cycle through all the keys on (up to) a 64-note keyboard, detecting in turn whether each key is depressed or not. The digital electronics in the synthesizer monitors line Z, and knows that a 'high' value at any given instant in the scanning cycle represents a specific pressed key. And, if we want to scan a wider keyboard, we just add more bits to the address lines. It's brilliant!

## Phew! Now Back To '70s Oberheims...

All the Oberheims sold from 1974 to 1977 were based upon monophonic modules called SEMs (Synthesizer Expander Modules). Each SEM offered two oscillators, a multi-mode filter, an LFO, two contour generators, and an amplifier (see Figure 8, page 158).

The 4-Voice had, as you may have guessed, four of these, each with CV and Gate inputs. This would suggest that, thanks to the digital scanning technique, it was possible to access each of the SEMs from the keyboard, as shown in Figure 9 (see page 158).

Of course, it's one thing to be able to say which keys are depressed at any given moment, but that's a far cry from generating the CVs, Gates and Triggers that tell a bunch of SEMs which notes to play. Figure 9 (on page 158) must be missing something...

The missing element is the so-called Voice Allocation Unit. You'll be pleased to know that the circuitry of this is too complex to discuss here, but the principle of its action is simple enough: it takes the note information generated by the decoder, checks to see whether there are any unused SEMs available, and then generates and sends the CV and Gate information to those SEMs. Of course, this isn't the end of the story. Matters are complicated considerably by a Split mode that divides the keyboard into two virtual keyboards, each with a predetermined number of SEMs allocated to them. Then there is Unison. This layers the voices on top of each other, reducing polyphony to just one note (ie. monophony).

But, ignoring these additional complications, let's consider the result of playing more notes than the synthesizer has voices. In the case of a 4-Voice, what happens when you press five notes simultaneously?

The answer to this depends upon the way in which the Voice Allocation Unit allocates the SEMs. For example, the 4-Voice offers one option in which it will cycle through each voice in turn. If you press just one note at a time, this means that the SEMs are allocated 1-2-3-4-1-2-3-4... and so on, as you play. But when you hold the first four notes, this system leads to 'note stealing' if you press a fifth key simultaneously (see Figure 10, right).

One way round this is to give earlier notes priority over later ones. This is the 'first-note priority' system discussed in Part 18 of this series (see *SOS* October 2000) but now applied polyphonically. Unfortunately, this will delay the onset of later notes, and the results may be even less desirable than note stealing (see Figure 11, right).

As you can see from Figure 9, the Oberheim 4-Voice was, in essence, four independent monosynths screwed into a wooden case together with a digitally scanned keyboard and its associated electronics. This meant that you had to set up all of the SEMs identically to play it as a conventional polyphonic instrument. Given the vagaries of this vintage of analogue electronics, this was all but impossible. SEMs are not famous for their stability, and getting four of them to stay in tune, correctly scaled, let alone with identical filter and contour characteristics... well, you could forget that! It's small wonder that few players took advantage of the synth's polyphonic potential. Indeed, some players just programmed each of the SEMs individually, and — in Unison mode — played the 4-Voice as one of the most overpowering monosynths of all time.

A year after its initial release in 1974, Oberheim added a 16-memory programmer to the 4-Voice, which should have solved the first problem (ie. setting the individual SEMs to the

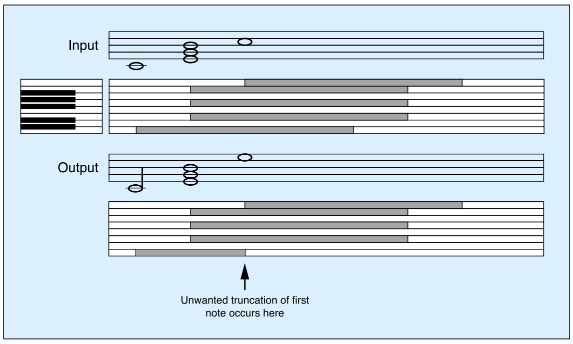


Figure 10: Note stealing on a 4-voice polysynth.

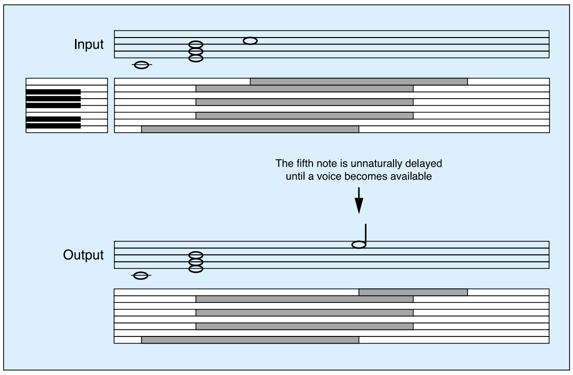

Input
The fifth note is unnaturally delayed until a voice becomes available
Output

Figure 11: Delayed notes on a first-note-priority 4-voice polysynth.

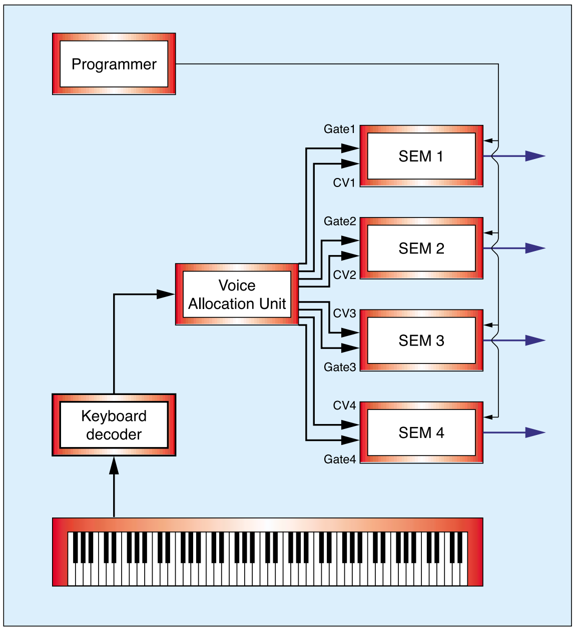


Figure 12: The basis of the Oberheim 4-Voice, showing the Voice Allocation Unit and programmer.

same sound for polyphonic use). Unfortunately, the programmer was unable to store and recall *all* of the parameters relating to a patch. Most seriously, it could not remember the resonance or the filter type selected.

The basic structure of the 4-Voice can be seen in Figure 12 (see page 160). Amazingly, the keyboard scanning and voice allocation functions are carried out entirely in hardware, with logic gates determining which keys have been pressed, which SEMs are available, and how the two should be paired together. In 1976, Emu picked up the baton of progress, took a huge leap for synthesis and adopted a microprocessor as the 'brain' of their 4060 keyboard — the same microprocessor that, in 1977, went on to become the basis for Sequential's Prophet 5.

But why stop there? If you'd worked out a mechanism for scanning the voltage status of a bunch of key switches ('high' or 'low') surely you could use the same mechanism to determine the states of front-panel switches such as voice selectors? Sure you could! For that matter, is there any reason why you couldn't then use it to measure voltages other than 0V or +5V? To put it another way, could you not use it to measure the settings of the knobs and faders on the synthesizer's control panel? You certainly could. Not only that but, using a suitable A-D converter, you could translate these values into a digital format suitable for storing in computer memory. And of course, people did. Blimey! This month, I've not only stumbled upon the secrets of practical polyphony, but also the way to design analogue synthesizers with memories.

So consider this... If a polyphonic synth allocates a limited number of voices to the notes as you play, it contains a significant amount of digital circuitry. Likewise, if a polyphonic synth has switches that select between preset patches, it contains a significant amount of digital circuitry. Similarly, if a polyphonic synth has user-programmable memories, it *too* contains a significant amount of digital circuitry. Therefore, a totally analogue polysynth can only be one that offers total polyphony provided by independent articulators for each note, and which has no presets and no memories. So here's this month's synth secret:

*In mainstream synthesis, there have only ever been two totally analogue polysynths: the Korg PS3100 and PS3300.*

## A Real Example

To finish this month's article, let's look at a real synth — the Sequential Circuits Prophet 600 — to illustrate all the above.

Released in 1983, this has six analogue voices that — unlike SEMs — are carefully calibrated to sound as close to identical as possible. However, the 600 uses a Z80 microprocessor to scan the keyboard, scan the control panel, and manage its 100 memories. The Z80 is even pressed into sound generation (as opposed to sound control) duties, calculating software-generated contours and LFOs that are applied to the synth's analogue VCOs, VCFs and VCAs. It all works like this...

Sequential Circuits' Prophet 600 synthesizer.

Every 200th of a second the Z80 calculates the values of the envelope generators, the LFOs and the effect (if any) of glide. It then refreshes the LEDs on the top panel, looks at either the pitch-bend or modulation wheel, refreshes a number of the internal control voltages, and then checks one (and only one) of the control panel knobs. Next, it scans the keyboard and, if you've played a note during that period, works out the voice assignments. This means that — depending upon the exact moment at which you press a key — your playing may be delayed by up to 5mS. You can also detect the consequences of the scanning and calculating when you adjust the 600's control knobs by small amounts. Listen carefully, and you can hear the effect as the processor jumps from value to value. This is one source of the famous 'digital zipper noise'.

Nowadays, keyboard scanning, note allocation, and numerous other functions are controlled by firmware within special chips called ASICs — Application Specific Integrated Circuits. You can think of these as dedicated microprocessors that have been pre-programmed to perform specific tasks). This is as true for the modern generation of analogue polysynths — for example, the Studio Electronics Omega 8 and the forthcoming Alesis Andromeda — as it is for the plethora of virtual analogue (VA) synths and digital workstation keyboards.

Thanks to Dave Smith, ex-head of Sequential Circuits, for supplying the photograph of the Sequential Prophet 600.

## Random Voice Assignment

**If you open a vintage polysynth such as an Oberheim OB8, you'll see on each voice board an LED that tells you whether this voice is playing or not. You can then press a succession of keys and see the LEDs march across the synth from 1 to 6 (or 1 to 8) before starting back at voice 1 again. If you hold a four-note chord, you can see, say, voices 1 to 4 light up, and then voices 5 to 8 (or whatever) cycle as before.**

**However, on a couple of early polysynths, the voices do not always rotate in such a strict, cyclic fashion, and this offers an unexpected benefit. Each of the voices in an analogue instrument will sound slightly different from the others — maybe with a different amount of detune, or with filters that are slightly more open or closed. These differences, if they are not too extreme, are a major source of the so-called 'organic' warmth of vintage polysynths.**

**Nevertheless, if the voices always play in the same order, you may occasionally hear a disturbing consistency as you perform, especially if you're playing a solo line. Let's suppose that voice three of a six-voice synth is slightly more 'open' than the others. If the voices speak in strict rotation, you'll hear your solo doing something like this: 'do-do-dee-do-do-do... do-do-dee-do-do-do...' This will place your performance firmly within electronic territory.**

**But if the synth's voices do not cycle in a predictable fashion, the same line may go: 'do-dee-do-do-dee-do... do-do-do-do-do-dee...' which will be much closer to the natural variations of tone and tuning of a 'real' musical performance.**

**Nowadays, of course, digital synths have 'analogue feel' parameters that add small random fluctuations to the sound, giving rise to much the same effect.**

# synth secrets

## PART 22: FROM SPRINGS, PLATES & BUCKETS TO PHYSICAL MODELLING

Onboard effects may seem like a relatively recent synth innovation, but even old modular synths offered analogue effects. Although they were basic, the freely patchable nature of modular synths allowed them to be used to create convincing acoustic instrument sounds — thus effectively physical modelling. **Gordon Reid** explains how.

Everyone knows what happens when you place an echo unit at the end of a signal chain: you play a note and, after the original, you hear a number of repetitions, usually dying away over the course of a few seconds. Similarly, if you place a reverberator at the end of the chain, your note is extended, with added 'ambience' — the perception that the note is produced in a room or hall of some sort.

In both these cases, you are using electronics to model acoustic spaces. In the former case, the model is that of one or two surfaces which create a handful of discrete echoes. In the latter, it's a model of a closed space, imitating the thousands of echoes that merge into the dense phenomenon we call reverberation.

If you were now to consider a collection of modern synths, you'd notice straight away that many of them incorporate electronic delay lines and reverberators immediately before their outputs, positioned there to enhance the sound produced by the sound-generating circuits (indeed, some manufacturers incorporated these effects to invigorate synths that would otherwise be dull and lifeless). But let's ask what would happen if you could take one of these effects — the reverb — and place it somewhat earlier in the signal path.

Sadly, few synthesists can try this for themselves, because the vast majority of modern synths do not allow you to reposition the effects. However, those of you lucky enough to own a full-blown modular synth can do this, as can those with the earliest integrated synths such as the VCS3 (1969) and ARP 2600 (1970).

I've added the release dates of those synths for a reason... in the 1960s, when synthesis was in its infancy, reverb was seen as far more than simply a way to add ambience to a signal. It paved the way for realistic emulations of acoustic instruments that you cannot imitate using basic VCO-VCF-VCA architectures. The simple reverb was the key to what we now call 'physical modelling'.

To understand how it was capable of doing this, we'll need to discuss what happens to a sound within a reverberant space, or — for that matter — within an analogue effects unit designed to emulate such a space. So let's start by revisiting what we already know about delays and reverb...

### From Delays...

Delays occur when sound reflects off solid objects. This phenomenon means that, in most circumstances, any sound produced by a point source will reach your ears from multiple directions (see Figure 1, right).

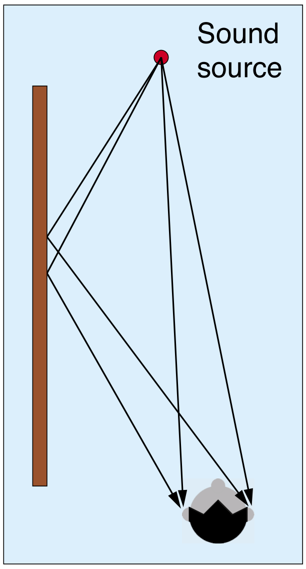


Figure 1: Delays off a hard surface.

If we think of this in terms of synthesizer modules, we can construct a patch (see Figure 2 below) that attempts to recreate the acoustic environment shown.

Of course, this is hugely oversimplified. For one thing, there's no such thing as a point source — anything moving the air must have an appreciable diameter. Secondly, we've ignored the reality of a wall that, far from being a perfect reflector, will diffuse the reflected sound, modifying its amplitude and tone. Thirdly, air is a less-than-perfect transmitter. To confirm this, stand at the back of a soggy crowd at Reading Festival on a windy day, and listen to the way in which the sound level rises and falls as the wind gusts (been there, done that). Fourthly, the dense matter between your ears conducts sound from one ear to the other. Clearly, Figure 2 is quite inadequate, even for the oversimplified example shown in Figure 1.

More importantly, we've overlooked the reflections from the floor. This adds at least four further signal paths. So let's now consider what happens when we place the sound source and listener in a closed acoustic environment such as a room... I've restricted Figure 3 (above right) to a handful of paths with a maximum of two reflections, but it's already

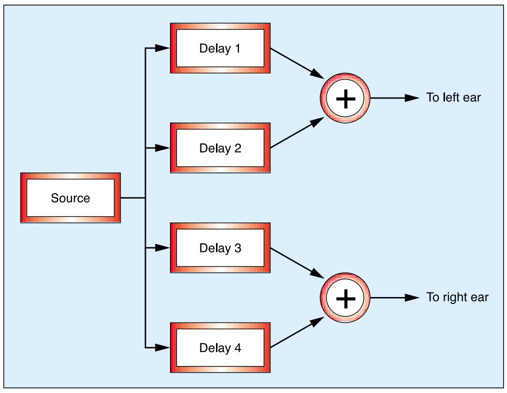


Figure 2: Using four delays to create the sound paths shown in Figure 1.

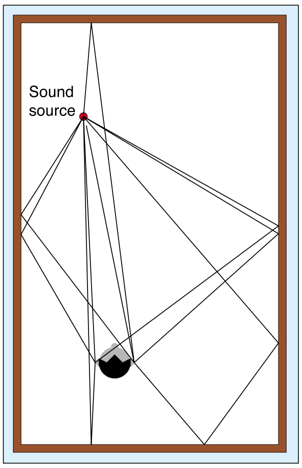


Figure 3: Some of the simplest wall reflections reaching the listener's ears (as seen from above).

getting complicated. Imagine how complex things become when we contemplate all the possible (and, therefore, inevitable) reflections off the walls, the ceiling and the floor. Indeed, if the walls are not acoustically absorbent, there's no reason why the sound should not bounce around hundreds or even thousands of times before it reaches your ears! Clearly, we can no longer think of this in terms of a handful of delay units...

## ...To Reverberation

Fortunately for those who like to look into these things, it's possible to analyse reverb by using a very short click (or 'impulse') to make sense of the sound reflections that occur in a closed space. If you were to do this, you'd find that there are three distinct temporal parts to the phenomenon (see Figure 4 overleaf). The first is the original click, while the second comprises the so-called 'early reflections'. These are the first, distinct, reflected sounds you hear — that is, the ones that bounce off just one or two of the available surfaces. The early reflections are rapidly followed by the thousands of reflected clicks that comprise the third region in Figure 4. Because the human brain is not usually capable of perceiving echoes separated by less than 30 milliseconds as distinct sounds (no matter how percussive the sound is), you hear the clicks in this third region as a composite sound — the 'reverb' portion of the reverbed click.

Because there's always something in a room that absorbs sound, natural reverberation dies away to silence, and we use a measure called the 'Reverberation Time' to quantify the rate at which it does so. This was defined by a chap named Wallace Sabine who, in the late 1890s, recognised that the difference in loudness between the human voice and the quietest sound you can hear is a factor of about 1,000,000. This difference is equivalent to approximately 60dB, so Wallace called his measure 'RT60' (see Figure 5, overleaf).

Fortunately (or not, when you try to do it) a simple formula allows you to calculate the theoretical value of RT60 for any given frequency for any given room. This can be your bathroom, your local rehearsal room or church hall, or even the Royal Albert Hall. You just need to know the size and absorbing properties of every material within it! Here is the equation:

$$RT60 = \frac{0.16V}{\sum (A \times S)}$$

In this formula:

- V is the volume of the room in cubic metres;
- A is the area of each type of absorber;
- S is the absorption coefficient of each type of absorber;
- and $\sum$ means 'the sum for all the different absorbers in the room'.

## Room Modes

At this point, you might think that you know most of what you need to understand reverberation. OK, it's fairly unlikely that Baroness Chumondsley-Smyth sitting in row three at the Royal Albert is going to cooperate when you ask to measure her soft absorbers, but, in principle, you're ready to calculate RT60 for any room.

Unfortunately, useful as RT60 may be, it only gives you information about the decay of the reverberation. This could be described as a 'time-domain' characteristic, as it tells you about how the reverb behaves over time — but to understand reverb more fully, and to see how it can be used in physical modelling, we must also consider its characteristics in the 'frequency domain'. In other words, we must consider the frequency response of the reverberation in a room.

To do this, let's remove the source and the listener from our reverberant room, and consider the properties of the room itself. Just as a stretched string has a fundamental mode of vibration plus harmonics (see part one of this series, way back in *SOS* May '99, or look at www.sospubs.co.uk/sos/may99/articles/synthsec.htm on the Internet) there are frequencies at which a rectangular room with reflective walls will 'resonate'. However, whereas a string is essentially 1-dimensional, the room is 3-dimensional, so there are many more permitted modes, governed by the little darling that is this equation:

$$F = \frac{c}{2}\sqrt{\left(\frac{m}{x}\right)^2 + \left(\frac{n}{y}\right)^2 + \left(\frac{p}{z}\right)^2}$$

In this formula:

- F is a resonant frequency;
- x, y, and z are the dimensions of the room in metres;
- c is the speed of sound in the room;
- and m, n and p are whole numbers.

This may seem complex, but in principle the room actually behaves very similarly to the string. The difference deriving from the three-dimensional nature of the room (as opposed to the string, which is to all intents and purposes one-dimensional) is

▶

ARP's 2600, though not fully modular, did allow the free patching of the spring reverb module in the signal chain, and would therefore be a good synth on which to try out some of the techniques expounded in this month's Synth Secrets.

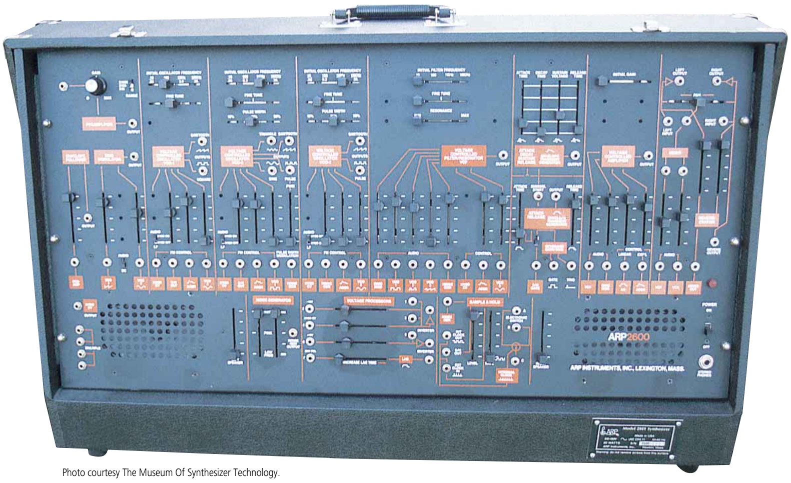


Photo courtesy The Museum Of Synthesizer Technology.

reflected in the three 'squared' terms in the equation (the equation for the string would have just one). Therefore, if we consider, say, the first 10 integers for each dimension, instead of having 10 harmonics, we obtain 1000 resonant modes!

Analysis shows that the frequency response of one of the simple families of these modes (n=0, p=0, and m=1 to infinity) looks like Figure 6 (right), where each peak along the frequency axis corresponds to a mode. As you can see, this is similar in concept, if not exactly in shape, to the comb filter we discussed in Part 4 of this series (see *SOS* August '99, or www.sospubs.co.uk/sos/aug99/articles/synthsecrets.htm).

Fortunately, rooms — even rectangular ones with hard, reflective walls — do not act as comb filters. This is because the thousands of modes are distributed unevenly throughout the spectrum, so the overall response is far flatter than Figure 6, with numerous smaller bumps and troughs. Nonetheless, a reverberant room has a definable frequency response.

Whether you choose to consider reverb in terms of its time-domain characteristics or in the frequency domain, you are simply considering two aspects of the same phenomenon. This equivalence of a room's reverberation characteristics and its frequency response is an illustration of what physicists often refer to as 'time/frequency duality'. We'll return to this point shortly...

There's just one thing to get across before we move on. It might seem as though moving the walls to change the room from rectangular to irregular would destroy room modes, but this is not the case. Sure, the modes will be distributed somewhat differently, leading to a changed frequency response, but an irregular room will, by and large, have a similar response to a regular one of the same volume. This will become an important consideration later in this article when we start to think about irregularly shaped musical instruments.

## From Theory To Practice

Right, that's enough acoustic theory — this is *Synth* Secrets, after all. Now let's turn our thoughts to music, and the ways in which reverb was recreated before the advent of cheap digital effects processors.

Given a large enough studio, a meaty enough power supply, and almost unlimited funds and patience, there's no reason why you couldn't create reverberation effects using thousands of delay modules and VCAs. But, obviously, it's impractical, even with an RT60 of just a few fractions of a second. What we need is a cheap and simple device that will produce the innumerable 'echoes' that comprise reverberation. Furthermore, it must make them die away in such a fashion that a sensible value for RT60 is obtained for the perceived 'resonant space' suggested by the delay of the unit.

The simplest way to do this is also the most impractical... you use a large, reverberant space called an 'echo chamber', and record your performance in this. If your sound is generated electronically rather than acoustically, you place a speaker at one end and a microphone at the other, play the 'dry' sound through the former, and re-record the 'wet' sound using the latter. In order to maximise the amount of reverb, you would not normally point the speaker at the microphone (see Figure 7, page 118).

A different reverb device requires a large steel plate, a suitable suspension, a large box, and a bunch of transducers. This is the plate reverb, a device of approximately the same size and weight — and a good proportion of the cost — of a grand piano. It uses a transducer to excite the plate, and one or more pickups that detect the reverberation created as the sound bounces around, reflecting off the edges (see Figure 8, page 118).

Unfortunately, neither of these mechanisms was small enough, light enough, or cheap enough to be placed within the early analogue synthesizers mentioned at the start of this article, so another device was needed. This, of course, was the spring reverb.

A simplistic view of the spring reverb suggests simply that you use a transducer to excite a spring, and a pickup to extract the reverberant sound at the other end (see Figure 9, page 118). In essence, this is true, but it is far from the whole story. Let's consider what's actually happening.

Let's say that the transducer excites the spring with an impulse.

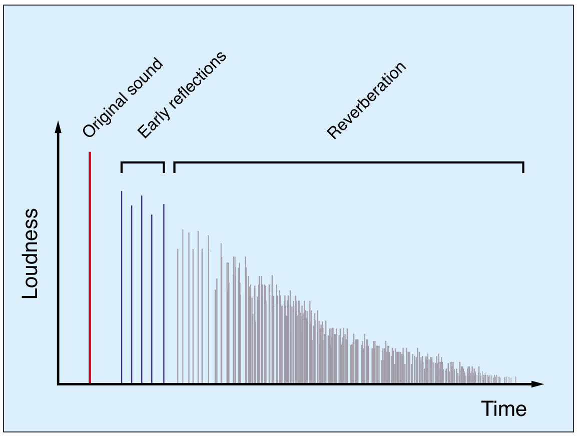


Figure 4: Reverberation.

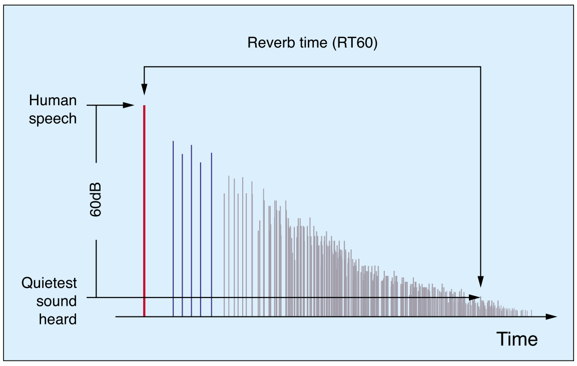


Figure 5: Defining the 'reverb time'.

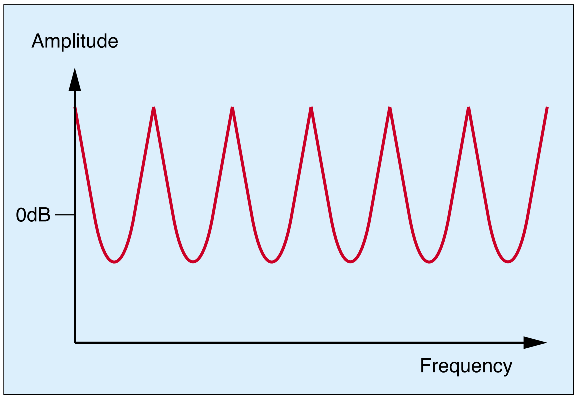


Figure 6: A family of modes in a rectangular room.

Ignoring the short springs at either end of the device (these are just shockmounts to isolate the reverb itself from outside influences), the wave thus produced travels up the spring to the pickup in a certain time. The output is then amplified and, usually, mixed with the original signal.

However, when the wave reaches the pickup, some of the energy is reflected back towards the transducer. Likewise, the depleted wave reaching the transducer is reflected back towards the pickup... and the cycle continues until all the energy is absorbed.

Let's now imagine that you are playing a 40Hz sine wave through this reverb, and that the transmission time from one end of the spring to the other is 12.5mS. The first delay reaches the pickup in 12.5mS. The reflected portion of the energy then travels back down the spring to reach the transducer 25mS after it left. However, the time taken for a sine wave of 40Hz to complete one cycle is also 25mS, so the next reflection reinforces the source. Furthermore, this happens time after time after time...

In contrast, a sine wave of 60Hz will arrive back at the transducer in anti-phase. Sure, the reflected energy is less than the energy being supplied by the transducer, but the two will, to some extent, cancel out. This means that a spring reverb has a frequency response similar to that shown in Figure 10 (right).

It looks familiar, doesn't it? It's that old time/frequency duality again, and it's very significant. Just as the reverberation bouncing around in a room exhibits a frequency response, so do the echoes travelling up and down a spring. So why doesn't a spring reverb have the characteristic warmth and depth of a concert hall or an echo chamber? The reason is simple... whereas the room has thousands of such modes, the spring has just three. The first is the longitudinal 'compression' mode. The second is the transverse wave exhibited by a stretched string... the spring moves up and down above and below its axis. The third is torsional... the spring twists and untwists along its length. Since each of these modes has different transmission speeds, the true frequency response of the spring is slightly smoother than Figure 10 would suggest. Nevertheless, the longitudinal mode dominates. Given the figures used above, you'll get echoes at 25mS, 75mS, 125mS... and so on, until the energy dissipates. This is very unlike true reverb, so a single spring reverb always exhibits a characteristic, metallic 'boinggg'.

Manufacturers have sought to improve on the single spring reverb by incorporating two or even three dissimilar springs in a single device. They have even gone so far as to assemble dissimilar sections into a single spring, and then use these composites to create more complex responses (see Figure 11, page 120).

Well, this is all fine and dandy, but hardly groundbreaking stuff. You use something that responds somewhat like a room to imitate the sound created by a room. Logical, or what?

But let's now return to synths, and consider the synthesis of acoustic musical instruments. Consider, for example, the body of a guitar, violin, contrabass, or any other hollow-bodied instrument. Ignoring the holes, these are all resonant spaces, like rooms, but with smaller dimensions. As for the holes, they're just like open windows: they let the sound out, but don't greatly reduce the amount of reverberation.

Now, it should be self-evident that the volume of an instrument's body, its shape, and the materials from which it is constructed will define its reverberant characteristics. In other words, the body will exhibit reverb, have a value for RT60 and, because of that ol' time/frequency duality, impose a frequency response upon the sound generated by whatever is exciting it. Research has shown that the vibration of a violin string at the bridge is remarkably close to being a sawtooth wave; and yet even if you ignore the years of practice required to develop an acceptable playing technique, a violin does not sound like a sawtooth wave. Clearly the body of the instrument plays a major part in shaping the sound that we recognise as that of a violin.

So here comes the leap of understanding towards which all the above leads... If we can recreate some of the reverberant frequency characteristics of a room using a spring reverb, can we emulate the modes (and therefore the frequency response) of a hollow-bodied instrument in the same way?

## 'Real' Synthesis

Figure 12 (see page 120) shows a basic representation of a simple, monophonic analogue synthesizer. In this, an

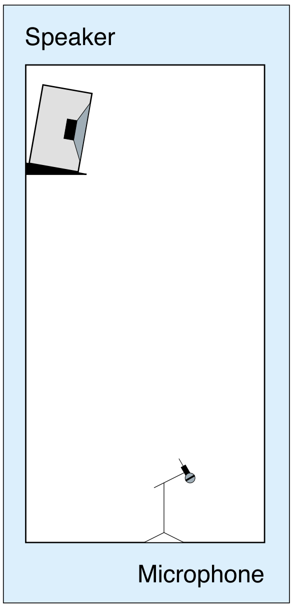


Figure 7: An echo chamber in a lift shaft.

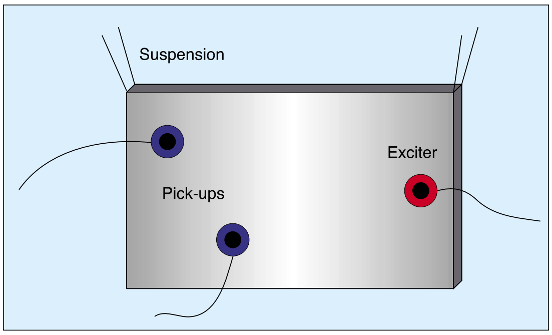


Figure 8: The important elements of a plate reverb.

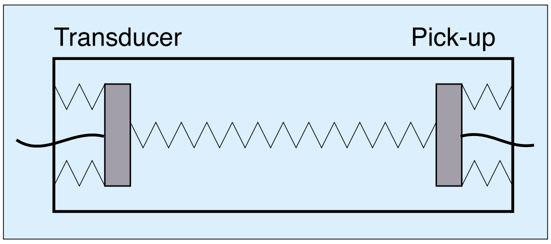


Figure 9: A 'single-spring' reverb.

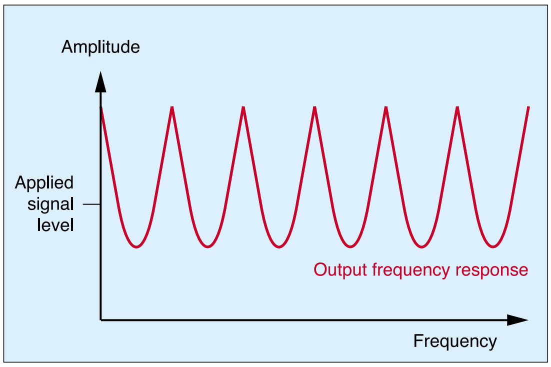


Figure 10: The frequency response of a single spring.

oscillator creates the basic sound, a contour-controlled low-pass filter modifies the tone in some way, a contour-controlled amplifier modifies the loudness in some way, and a low-frequency oscillator introduces one or more forms of modulation. Finally, a reverb unit adds 'life' and 'air' to the resulting sound.

Flexible though this structure is, its low-pass filter cannot recreate the tonal quality of the modes exhibited by a reverberant space. Therefore, if you try to imitate a violin or acoustic guitar, the result is unconvincing. But let's now move the reverb 'inside' the patch (see Figure 13, right).

In this configuration, the reverb imposes its complex frequency response upon the output from the oscillator, emphasising some harmonics while suppressing others. Therefore, as you play up and down the keyboard, the characters of the individual notes change, much like those of an acoustic instrument.

Unfortunately, the reverb time of our spring reverb is far too long to emulate a real instrument. While it's not too shabby at recreating room modes, the spring would synthesize a 'violin' with a cavity over four metres long! Smaller springs would, for obvious reasons, produce more appropriate modes, and it is for this reason, perhaps, that early synthesists were able to use the six-inch springs in their instruments in this way. Nevertheless, we're not quite where we need to be, which is in the delay range of about 1mS to 4mS.

So now we turn to a device invented in 1969 by John Sangster at Philips, and introduced into affordable music technology in the mid-'70s: the Bucket Brigade Device. The BBD is a series of transistors connected in such a way that, if you present an analogue signal to the input, the signal comes out the other end, slightly delayed, slightly distorted but still recognisably the same signal. The name comes from the image of firemen passing buckets of water up a human line to quench a fire. Think about sampling the fluctuating voltage of an audio signal, and then passing each voltage down the line in a bucket... you get the picture.

You can chain BBDs to increase their delay times, and modulate the speed at which the 'sample buckets' are passed from one stage to the next. This makes them ideal for analogue effects units such as choruses, phasers, and flangers. They are also ideal for use as comb filters (as you may remember from part four of this series, comb filtering results when you combine two otherwise identical signals when one is very slightly delayed with respect to the other). But when placed in circuits with audio feedback to create reverb effects, it is the BBD's ability to produce very short delays that is important to us.

This, finally, explains how we can achieve physical modelling by placing a reverb with a suitable value for RT60 within the signal-generating architecture of a synth. Far from simply adding ambience to a previously generated signal, the short delay times generated by a BBD reverb will — just like the body of a violin — superimpose frequency characteristics reminiscent of an acoustic space upon basic signals such as a sawtooth wave.

Of course, with a single BBD reverb, we're still limited to a single dimension. So let's add another two parallel reverbs to our signal path, and mix the results before passing them to the rest of the synth. Now we're getting somewhere... provided that the reverb times are different for each of the BBDs, we will obtain three families of modes, making the response more '3-dimensional' in its effect (see Figure 14, right).

We can now tailor each BBD to generate an approximation of the dimensions of the acoustic 'body' we desire (remember... an irregular space will have a similar response to a regular one of the same volume). We can even adjust RT60 (normally called the 'reverb time') to determine whether our 'virtual instrument' is made of a hard substance or a softer, more absorbent material.

Unfortunately, I doubt that you'll be able to test this with your latest digital workstation, because it's unlikely that it will allow you to place its reverbs at the correct point within the signal chain.

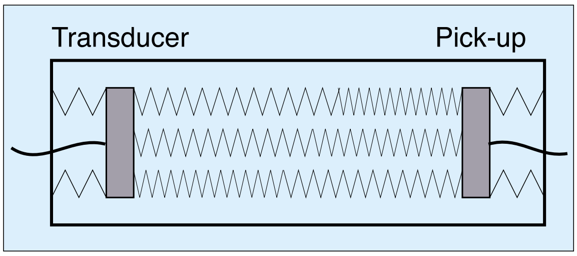


Figure 11: A three-spring assembly using dissimilar elements in each.

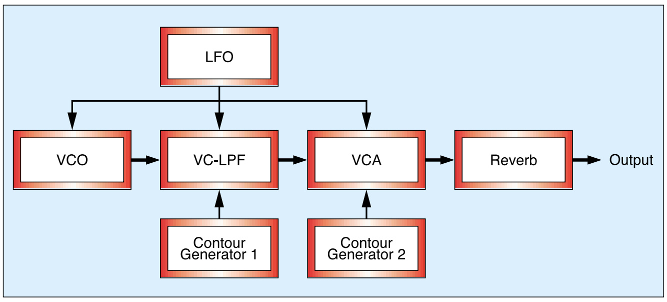


Figure 12: A simple monosynth structure.

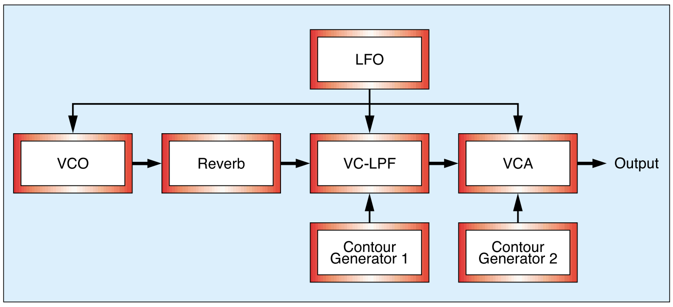


Figure 13: Using a spring reverb to imitate the 'modes' of an acoustic space.

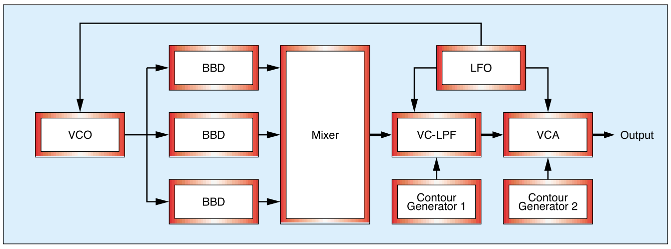


Figure 14: Modelling an acoustic space using analogue modules.

I would like to thank Christopher Hicks and Dave Betts of CEDAR Audio who, once again, have tried to ensure that I don't end up with physically modelled egg on my face. Any remaining cock-ups are entirely my own.

Photo: Mark Ewing.

Not many modern synths have the signal-routing flexibility to allow the kind of experimentation described in this month's article, but Clavia's fantastic Nord Modular, whose signal routing is freely patchable in software, is one such instrument.

However, if you have a genuine modular synth that incorporates a number of BBD delays/comb filters, you'll be able to create some remarkably authentic acoustic instrument sounds (this is also true with the more modern 'virtual modulars', such as the Nord Modular shown opposite, because these too offer short delay lines). Sure, you'll have to think about the size of the instrument you want to emulate and then calculate the appropriate delay times, but that's no big deal. In fact, you can do this with a sharpened pencil and just a little GCSE maths.

To prove this, let's take the equation on page 115 and say that we want to calculate a one-dimensional mode of an instrument with a lowest resonance at, for example, 500Hz. Given that the speed of sound is approximately 340 metres per second, we obtain the following relationship:

$$500 = \frac{340}{2}\sqrt{\left(\frac{1}{x}\right)^2}$$

If you manipulate this a little, you'll find that x (the dimension of the cavity) is 340/1000, which is 0.34m, or about 13 inches. Furthermore, since the speed of sound is 340 metres per second, and the sound has to travel from one end of the cavity to the other, and then back again... it follows that the delay time needed is two microseconds, exactly as we would expect for a resonance at 500Hz.

So here's this month's Synth Secret:

*You don't need powerful DSPs to dabble with physical modelling of acoustic spaces... a few analogue reverbs are more than enough.*

Of course, once you've emulated your acoustic instrument, you'll want to place *another* reverb at the end of the signal chain, just to place the sound in a pleasing ambient 'space'. Indeed, you'll probably choose a huge, digital hall reverb algorithm with a slow early-reflections setting and an RT60 of many seconds, because acoustic instruments often sound best in 'concert-hall' sized spaces. It just goes to show that, as always, there's more to this synthesis lark than you might first imagine.

Finally, I'm going to leave you with a bit of a teaser... This month's entire Synth Secrets has dealt with the duality of reverberation and the frequency responses of closed (-ish) acoustic spaces. But couldn't we have avoided this talk of echoes, RT60s, room modes, and all that other stuff, and achieved the same result with a bunch of fixed (or 'formant') filters? That's what we'll discuss next month...

# synth secrets

## PART 23: FORMANT SYNTHESIS

Ever heard a synth talk? If you have, it's due to formant synthesis. **Gordon Reid** concludes the purely theoretical part of this series with a look at the theory of analogue formant synthesis, how it relates to the human voice, and modern digital synths like Yamaha's FS1R.

Last month, we discussed a way of emulating acoustic musical instruments using short delay lines such as spring reverbs and analogue reverb/echo units. At the end of that article, I posed the following question: "Couldn't we have avoided this talk of echoes, RT60s, room modes, and all that other stuff, and achieved the same result with a bunch of fixed (or 'formant') filters?". This month, we're going to answer that question.

### A Little More On Filters

Let's start by remembering the four types of filters that I first described in Parts 4, 5 and 6 of this series (*SOS* August to October '99). These are the low-pass filter, the high-pass filter, the band-reject filter, and the band-pass filter (see Figures 1(a)-(d), right). OK, we're all sick to the back teeth of descriptions of the low-pass and high-pass filters in conventional synthesis. No matter. This month we're going to concentrate on the band-pass filter (or BPF), so read on...

Imagine that we place several of the band-pass filters shown in 1(d) in series (see Figure 2, below). It should be obvious that at the peak, the signal remains unaffected. After all, a gain change of 0dB remains a gain change of 0dB no matter how many times you apply it. But in the 'skirt' of the filter, the gain drops increasingly quickly as you add more filters to the signal path (see Figures 3(a) and 3(b), right).

As you can see, the band-pass region of the combined filter tightens up considerably as you add more filters to the series. This, of course, is directly analogous to the situation wherein we add more poles to a low-pass filter circuit to increase the cut-off rate from 6dB/octave to 12dB/octave, 18dB/octave, 24dB/octave... and so on.

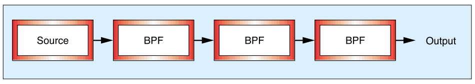
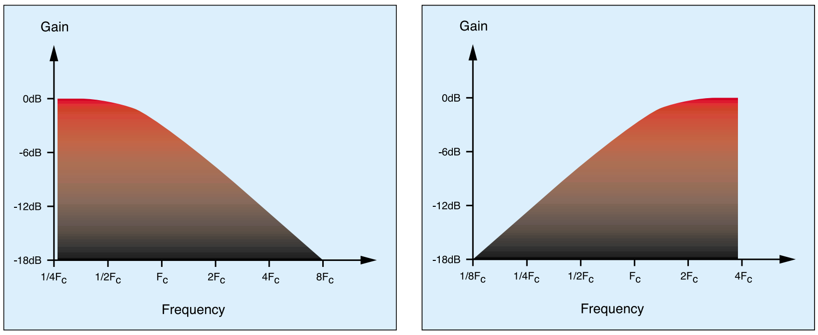
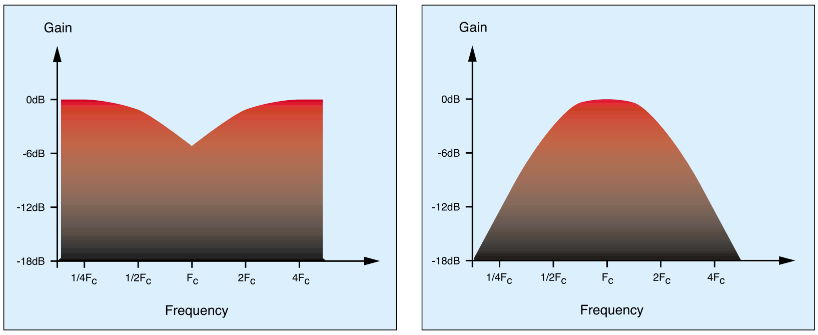


Figure 2: Placing a number of band-pass filters in series.

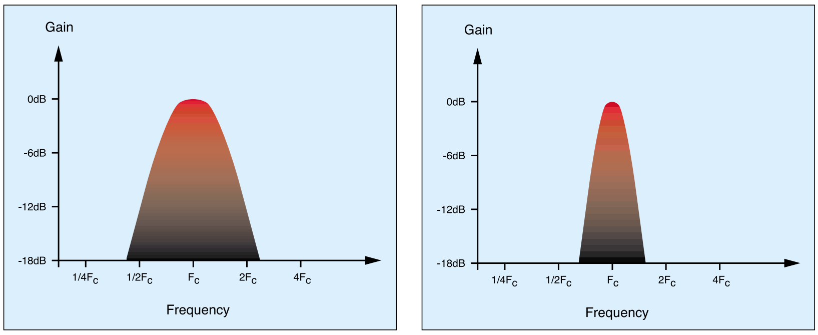
Now let's look at the case in which we place the band-pass filters in parallel rather than in series (see Figure 4 opposite). If we set the centre frequency 'F$_c$' of all the filters to be the same, the result is no different from using a single filter and, depending upon the make-up gain in the mixer, will look like Figure 1(d). Much more interesting is



Figures 1(a) and 1(b): 6dB/octave (one-pole) low-pass and high pass filters.



Figures 1(c) and 1(d): Band-reject and band-pass filters.



Figures 3(a) and 3(b): The responses of placing two one-pole band-pass filters in series, and of placing four one-pole band-pass filters in series.

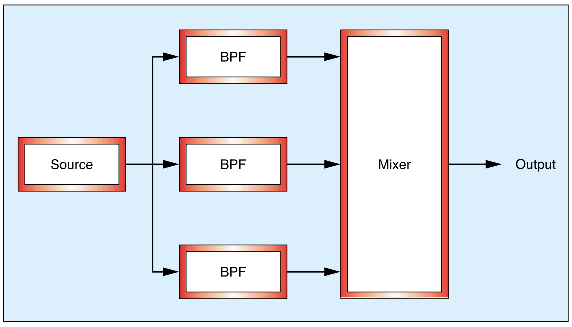


Figure 4: Configuring a number of band-pass filters in parallel.

the response when you set $F_c$ to be different for each filter. You then obtain the result shown in Figure 6(a), on page 120. As you can see, the broad responses overlap considerably, giving rise to a frequency response with many small bumps in the pass-band (the pass-band is so-called because it is the range of frequencies in which the signal can pass relatively unaffected through the system). Outside the pass-band, the gain tapers off until, at the extremes, little signal passes.

Let's now add more BPFs to each signal path (see Figure 5, page 120) to sharpen the responses. Provided that the centre frequencies of each of the filters in a given path are the same (I've labelled these f1, f2 and f3) but are different from those of the other paths, we obtain three distinct peaks in the spectrum, as shown in Figure 6(b). The filters severely attenuate any signal lying outside these narrow pass-bands, and the output takes on a distinctive new character.

## A Little More On Modelling

If the frequency response in Figure 6(b) looks familiar, let me refer you back to last month's *SOS* and, in particular, the diagram reproduced here as Figure 7 (also on page 120). This is the frequency response of a set of 'modes' which, as we discovered last month, may be the result of passing the signal through a delay unit such as a spring reverb or bucket-brigade device. It also represents a single set of the 'room modes' that arise as a consequence of the sound bouncing around inside a reverberant chamber.

OK, so Figures 6(b) and 7 look rather different, but it's not too difficult to design and configure a set of band-pass filters that responds similarly to Figure 7. You may need to add a 'bypass' path to ensure that an appropriate amount of signal passes between pass-bands, but we're going to ignore that. Furthermore, if the $F_c$s are precise (and this is yet another instance of the phenomenon known as time/frequency duality, discussed last month) the filter-bank will impose the appropriate set of delays upon any signal passed through it.

You may also recall that, last month, we used three delay units to achieve a superficial emulation of a three-dimensional space. We then shortened the delay times of each unit until the dimensions of our 'virtual' reverberant chamber were no larger than the body of a guitar or violin. It's a small leap of intuition, therefore, to realise that we could use three banks of band-pass filters to achieve the same effect (see Figure 8, page 120).

The most important thing about this configuration is that the frequency response of the filter banks, and the timbre that they therefore impose on the signal, is independent of the pitch of the source. To see how this differs from conventional synthesizer filtering (in which the filter cutoff frequency often tracks the pitch of the note being played) consider Figures 9 and 10 on page 122.

The first of these shows the spectral structure of a 100Hz sawtooth wave played through a set of band-pass filters with $F_c$s of 400Hz, 800Hz and 1200Hz. As you can see, the harmonics that lie at 400Hz, 800Hz and 1200Hz are amplified in relation to the rest of the spectrum, thus emphasising (in this case) the fourth, eighth and 12th harmonics.

The second uses exactly the same filter bank but, because the source signal has its fundamental at 200Hz (ie. one octave higher) it's the third, seventh, and 11th harmonics that are exaggerated. As one would expect, this changes the character of the sound considerably.

At this point, you may be asking why you can't use a graphic equaliser (or a multi-band fixed filter such as the one shown in Figure 10, on page 122) to create these distinctive peaks in your sounds' spectra. The reason for this is simple: the peaks of the equalisers that comprise a conventional filter-bank are too broad, so each filter boosts a wide range of frequencies. This is in sharp contrast to room modes and instrument modes which are, well... sharp (see Figure 11, page 122).

Clearly, we need something more specialised if we are to model cavity modes using filters. But before we come to this, let's expand our horizons beyond simple peaks of fixed frequencies and fixed gains. We need to consider...

## Things That Make You Go "Hmm" (Well, "Aaah" Anyway)

Let's ask ourselves what happens when the spectral peaks in the signal are less regular — ie. not evenly spaced, not all of equal gain, and not all of equal width. Furthermore, can we describe what happens when their positions (their $F_c$s) are not constant? To investigate this, we're going to consider the most flexible musical instrument and synthesizer of them all... the human voice.

Because you share the basic format of your sound production system with about six billion other bipedal mammals, it's safe to say that all human vocalisations — whatever the language, accent, age or gender — share certain acoustic properties. To be specific, we all push air over our vocal cords to generate a pitched signal with a definable fundamental and multiple harmonics. We can all tighten and relax these cords to change the pitch of this signal. Furthermore, we can all produce vocal noise. The pitched sounds are generated deep in our larynx, so they must pass through our throats, mouths, and noses before they reach the outside world through our lips and nostrils. And, like any other cavity, this 'vocal tract' exhibits resonant modes that emphasise some frequencies while suppressing others. In other words, the human vocal system comprises a pitch-controlled oscillator, a noise generator, and a set of band-pass filters! The resonances of the vocal tract, and the spectral peaks that they produce, are called 'formants', a word derived from the Latin 'formare', 'to shape'.

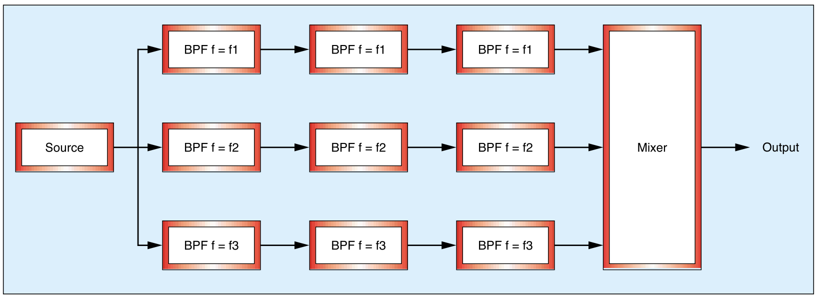


Figure 5: Adding BPFs to create a sharper filter response in each signal path.

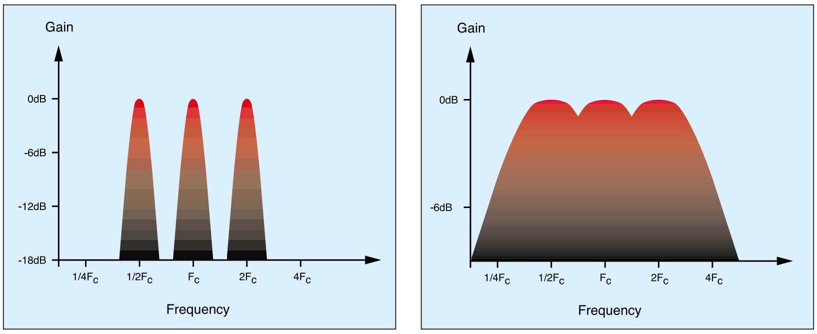


Figure 6(a) and 6(b): Configuring (a) single-pole BPFs in parallel, and (b) multi-pole BPFs in parallel.

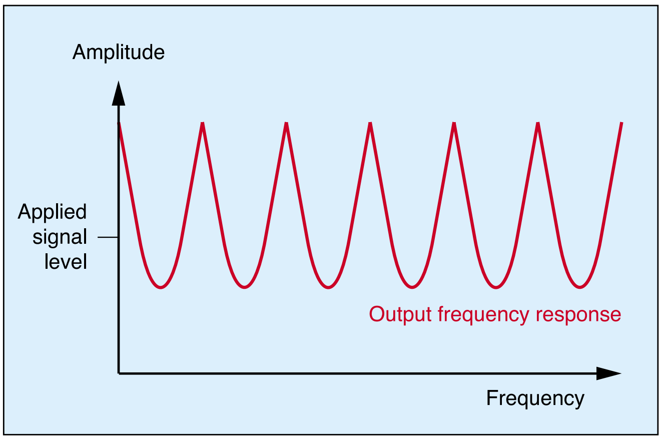


Figure 7: A family of modes.

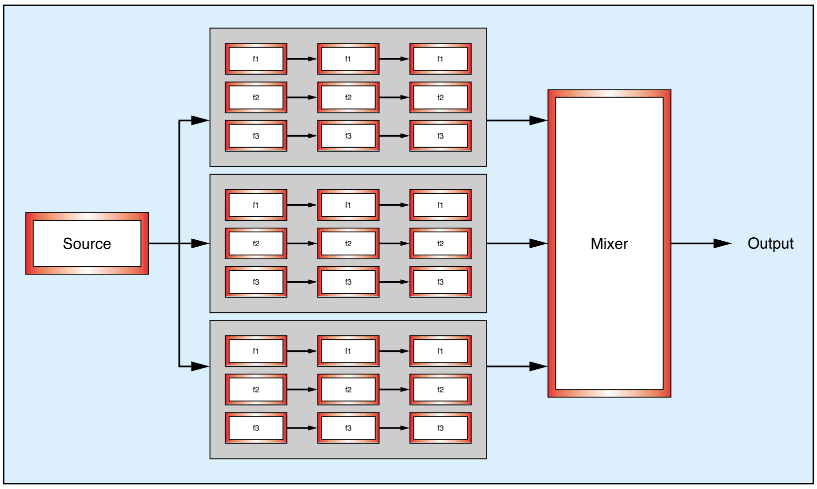


Figure 8: Using multiple band-pass filters to create three families of modes.

Measurement and acoustic theory have demonstrated that the centre frequencies of these formants are related to simple anatomical properties such as the length and cross-section of the tube of air that comprises the vocal tract. And, since longer tubes have lower fundamentals than shorter ones, it's a fair generalisation to suppose that adult human males will have deeper voices than adult human females or human children.

Now, ignoring the sounds of consonants for a moment, it's the formants that make it possible for us to differentiate different vowel sounds from one another (consonants are, to a large degree, noise bursts shaped by the tongue and lips, and we can model these using amplitude contours rather than spectral shapes). The following table shows the first three formant frequencies ($F_c$s) for a range of common English vowels spoken by an adult male. Note that, unlike many of the characteristics we have discussed in the past 22 instalments of Synth Secrets, these do not follow any recognisable harmonic series. Nor do they conform to series defined by Bessel functions. This is yet another reason why the filters within graphic EQs and fixed filter banks are unsuitable. Such filters tend to be spaced regularly in octaves or fractions of octaves, whereas formants are distributed in seemingly random fashion throughout the spectrum.

| VOWEL SOUND | AS IN... | F1 | F2 | F3 |
|---|---|---|---|---|
| "ee" | leap | 270 | 2300 | 3000 |
| "oo" | loop | 300 | 870 | 2250 |
| "i" | lip | 400 | 2000 | 2550 |
| "e" | let | 530 | 1850 | 2500 |
| "u" | lug | 640 | 1200 | 2400 |
| "a" | lap | 660 | 1700 | 2400 |

Given just these three frequencies you can, with precise filters, create passable imitations of these vowel sounds. This is because (as demonstrated by experiments as long ago as the 1950s) your ears can differentiate one vowel from another with only the first three formants present. So — provided that you use an oscillator with a rich harmonic spectrum — you can patch a modular analogue synthesizer to say "eeeeeeeeee" (as shown in Figure 12, on page 124). If you have almost unlimited funds and space, plus a particularly chunky power supply, you can add more formants to make the resulting sound more 'human'. With six or more formants, the results can be very lifelike indeed.

Mind you, in real life, things are far from this simple. Every human vocal tract is different from every other, so the exact positions of the formants differ from person to person. In addition, the amplitudes of the formants are not equal, and the widths of the formants (expressed as 'Q') vary from person to person, and from sound to sound.

Although it's tempting to shy away from mathematical expressions, Q is a simple way to express and understand the sharpness of a band-pass filter or formant, and is defined in the following formula:

$$Q = \frac{\text{Resonance (in Hz)}}{\text{0.5 x EQ width (in Hz) at half maximum Gain}}$$

This states that you calculate the quantity 'Q' by dividing the centre frequency of the curve (in Hz) by the half-width of the EQ curve (measured at half the maximum gain).

Let's take, for example, an EQ curve with a centre frequency of 1kHz and a width (at half the maximum gain) of 200Hz. The Q would therefore be 1,000/100, which is 10. Similarly, if the centre frequency remained at 1kHz but the width was just 50Hz (a shape represented by the blue curves in Figure 11) the Q would be 40 — a very 'tight' response indeed.

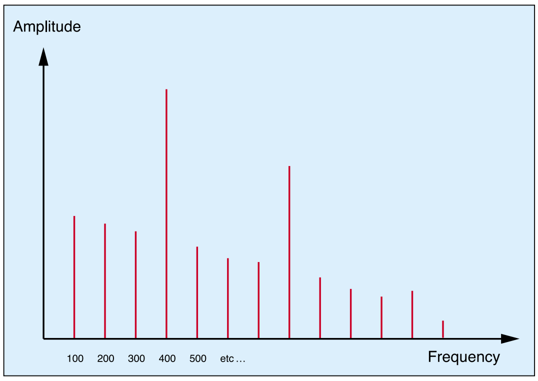


Figure 9: A 100Hz signal played through a system with $F_c$s at 400Hz, 800Hz, and 1200Hz.

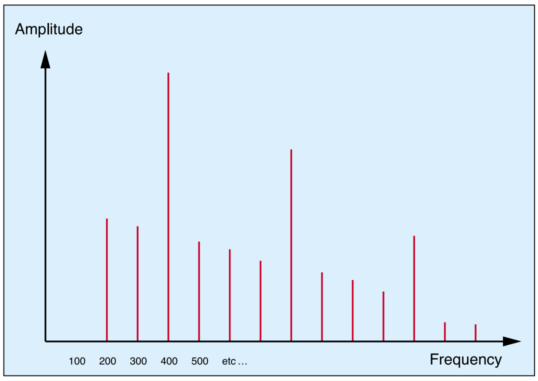


Figure 10: A 200Hz signal played through a system with $F_c$s at 400Hz, 800Hz, and 1200Hz.

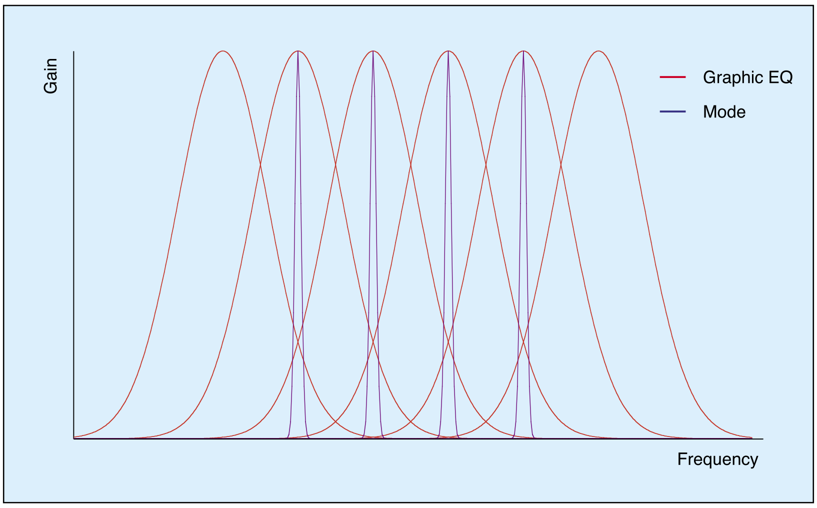
Clearly, the sharper the EQ curve or formant, and the fewer frequencies that it affects in any significant fashion, the higher the quantity 'Q' becomes. Conversely, if the curve is very broad (the red curves in Figure 11) and significantly affects a wide range of frequencies, the Q is low.

Understanding this, we can extend the above table, adding amplitude information to create formants that are more accurate. Let's take "ee" as an example...

| | VOWEL SOUND "EE" | GAIN (dB) | Q |
|---|---|---|---|
| F1 | 270 | 0 | 5 |
| F2 | 2300 | -15 | 20 |
| F3 | 3000 | -9 | 50 |

The mathematically inclined among you may have noticed that these Qs (which increase with $F_c$) suggest a bandwidth for all the formants of around 100Hz. This is indeed the case for a man's voice, ▶

Figure 11: The EQ curves of graphic equalisers are not suitable for creating tight peaks in the sound spectrum.



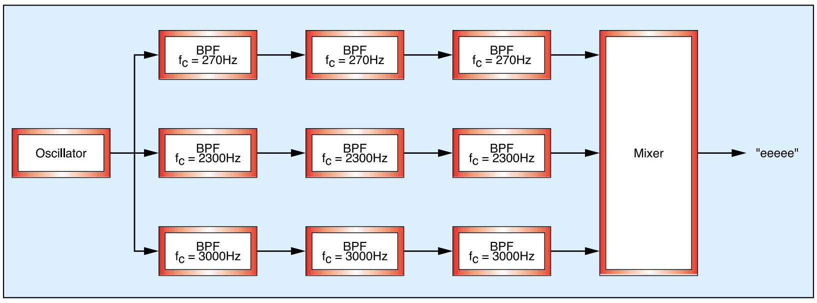


Figure 12: Say "eeeeeeeeee".

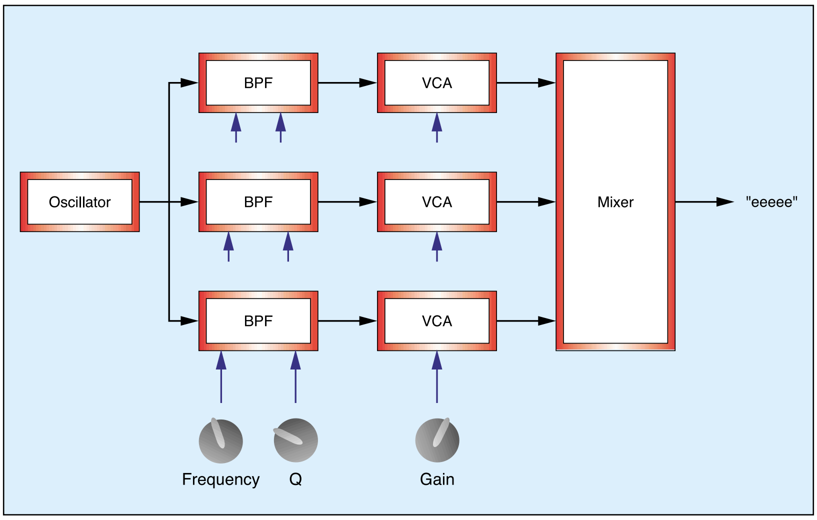


Figure 13: Say "eeeeeeeeee" more like a human, please.

although the bandwidth increases somewhat with formant frequency. Women's formants are — as a rule — slightly wider than men's (no sniggering at the back, please).

Therefore, to make our vocal synthesizer more authentic, we must make the Qs of our band-pass filters controllable, and add a set of VCAs, as shown in Figure 13 (page 124). Having done this, we're perfectly justified in calling our configuration a 'formant synthesizer'.

Unfortunately, this still isn't the end of the story, because the sound generated in Figure 13 is static, whereas human vowel sounds are not. We need to make the band-pass filters controllable, applying CVs to the filter $F_c$s and the VCAs' gains. If we analyse human speech, we find that the second formant is often the one that moves most, which suggests that this the most important clue to understanding speech. Furthermore, we would discover that the relative gains of the formants can swap... sometimes the lowest formant is the loudest, and sometimes it's the second or third.

Knowing all this suggests a novel approach to speech transmission — at least for vowel sounds. Instead of transmitting 44,100 16-bit samples per second down a line of limited bandwidth, we could send a handful of parameters — the formant frequencies, gains and Qs — once every few milliseconds, and reconstruct the voice at the other end of the line. If we restrict ourselves to, say, six formants, and update the parameters 100 times per second, we would require, at most, 1800 words of information, cutting the required bandwidth by a factor of almost 25.

Unfortunately, interesting as this is, I can see *Sound On Sound*'s editors glowering at me from the wings. This is because, if we proceed any further down this route, we'll find ourselves firmly within speech recognition and resynthesis territory, and that's a step too far for Synth Secrets.

## Practical Formant Synthesis

Just as the precise positions and shapes of the formants in a human voice allow you to recognise the identity of the speaker as well as the vowel sound spoken, the exact natures of the static formants make the timbres of a family of acoustic instruments consistent and recognisable from one instrument to the next, and over a wide range of played pitches. This is down to simple mechanics. For example, all Spanish guitars are of similar shape, size, and construction, so they possess similar formants and exhibit a consistent tonality that your ears can distinguish from say, a plucked viola playing the same pitch. It therefore follows that imitating these formants is a big step forward in realistic synthesis.

Of course, you may not have access to the room full of modules demanded by the practical configuration of Figure 13, so let's ask whether there are any simpler and cheaper ways to experiment with formant synthesis.

Firstly, although I discounted graphic EQs and fixed filters earlier in this article, there is a common type of equaliser that is quite suitable for imitating fixed formants. This is the parametric equaliser, which typically offers three controls per EQ: the centre frequency (often called the 'resonant' frequency), the gain, and the 'Q'. These, as you can see, are exactly the controls needed to perform as required in Figure 13. Sure, they're not dynamically changeable, but that's an unnecessary facility if we wish to synthesize hollow-bodied instruments such as guitars and the family of orchestral strings.

In principle, you can set up a parametric EQ to impart the tonal qualities of families of instruments. This then allows you to adjust other parts of the synthesizer — such as the source waveform, a low-pass filter, or brightness and loudness contours — to fine-tune the 'virtual instrument'. For example, you could filter the waveform and shorten the contours' decays to swap between the sound of bright new guitar strings, and the dull 'thunk' that emanates from the 10-year-old rubber bands on my aged Eko 12-string.

But what if you want to make your synth talk? While fixed formants are sufficient for synthesizing fixed-cavity instruments, they are inadequate for vocal synthesis. We need something more powerful than a parametric EQ...

Consider the resonant multi-mode filter shown in Figure 14, right. This offers a band-pass mode with CV control of frequency ($F_c$), manual control of Level

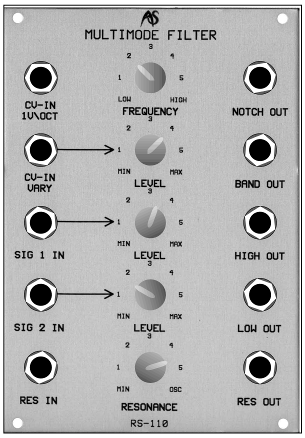


Figure 14: A resonant multi-mode filter.

(Gain) and (if coupled to a VCA) CV control of resonance (Q). If used alongside a second VCA that provides dynamic control over amplitude, this could be a satisfactory basis for a 'formant synthesizer'.

Unfortunately, you will require three modules for each formant, with appropriate CVs for each (see Figure 15, on page 126). Although you only need one set of formant filters (remember, the formants remain constant for all pitches), you'll need to be able to supply all the CVs to change the sound as the note progresses. Furthermore, since Figure 15 depicts a monophonic instrument, the complexity will increase considerably if you wish to add polyphony. This can lead rapidly to an enormous and unwieldy monstrosity. Nonetheless, there's no reason why it should not produce, say, a recognisable "aaaa" sound which, by suitable manipulation of the CVs, changes smoothly to an "eeee" sound.

## But The Sensible Solution Is...

We've encountered this spiralling complexity once before in Synth Secrets. It was when we discussed a polyphonic analogue FM synthesizer. While possible, this proved to be totally impractical, and the solution was found in the digital FM technology pioneered by Yamaha and incorporated to such devastating commercial effect in the DX7.

Well, we're going to the same source for the solution to the complexities of formant synthesis. While it has had far less impact than the DX7, Yamaha's FS1R (reviewed *SOS* December '98) is a superb and under-rated synthesizer designed specifically to imitate the moving formants found in speech, as well as the fixed-frequency formants of acoustic instruments. It even incorporates unpitched operators that can imitate consonants, as well as produce percussion and drum sounds. With real-time processing of the formants' $F_c$, Gain, Q, and a parameter called 'skirt', it is quite capable of emulating words and phrases — something that you would need a huge assembly of analogue modules to achieve.

Finally, let's take a look at how the FS1R imitates the frequency response of a harmonically rich signal (or noise) passed through a resonant low-pass analogue filter (see Figure 16, above right). Yes, yes... we've seen it all before, but bear with me one more time.

Surprisingly, we can reconstruct this frequency response using just two formants — one with a centre frequency of 0Hz and a Q of, say 0.1, and one with a centre frequency equal to the analogue filter's $F_c$, and with a Q of, say, 10 (see Figure 17, right).

The result is remarkable. What's more, we can make the formant-generated sound respond very similarly to the analogue case. To be specific, we can shift the perceived cutoff frequency by moving the centre frequency of the upper formant while narrowing the Q of the lower formant by an appropriate amount. Do this in real time, and you have a sweepable filter. Furthermore, we can increase and decrease the perceived resonance by increasing or decreasing the amplitude of the upper formant alone.

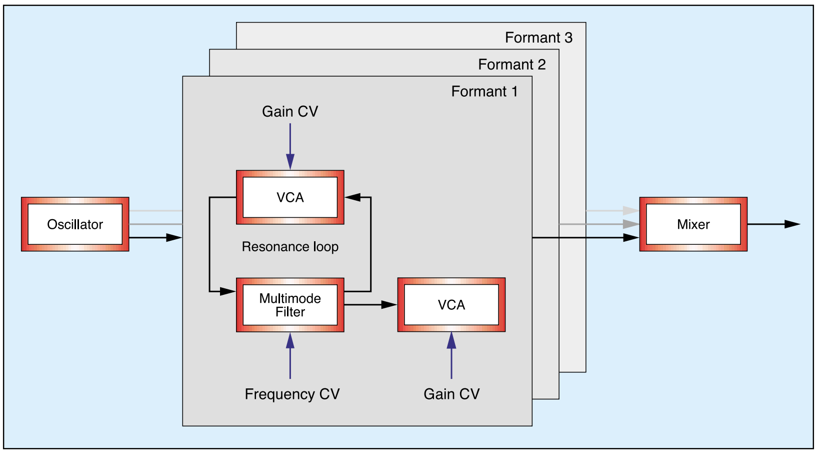


Figure 15: Part of a formant synthesizer constructed from multi-mode filters.

So, as has happened so many times before, we've come full circle. Analogue and digital synthesis — in this case, digital formant synthesis — are simply different ways of achieving similar results. So what else is new?

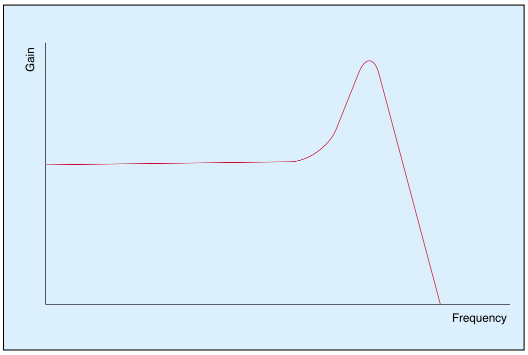


Figure 16: A typical frequency response from a resonant low-pass filter.

## Footnote

Next month we reach Synth Secrets 24... which means we've gone through nearly two years of investigation into some of the fundamentals of sound and synthesis. Not surprisingly, we've covered many of the major aspects of the subject, so it's time to ask, "is this end of our odyssey?".

Not a bit of it. However, from now on, we're going to turn things on their head. Instead of delving into some esoteric aspect of acoustics or electronics and seeing where it takes us, we're going to select a family of orchestral instruments and see how we can emulate them using our synthesizers. In other words, we're finally going to get our hands dirty with a bit of genuine synthesis. And where there's muck, there's brass...

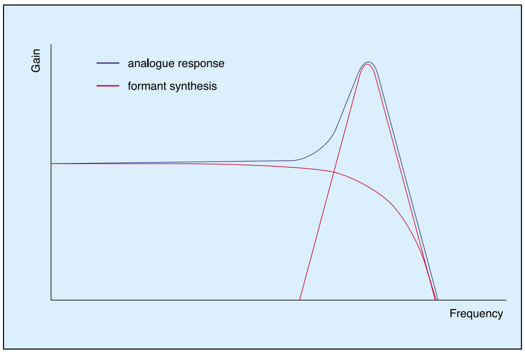


Figure 17: Imitating a resonant low-pass filter using two formants.

Yamaha's FS1R — a very powerful digital formant synthesizer.

# synth secrets

## PART 24: SYNTHESIZING WIND INSTRUMENTS

**Gordon Reid** embarks on a journey to synthesize convincing woodwind and brass. This month, he considers how these instruments make their sounds in real life.

At the end of the last Synth Secrets, I promised that we would start looking at the synthesis of traditional instruments. And that's what we're going to do this month, starting with wind instruments. But don't imagine for a moment that I'm going to draw a couple of block diagrams, tell you to set this voltage-controlled wotsit to that value, and spoon-feed you a couple of simple synth patches. After all, if you understand the nature of a sound, you can make huge leaps forward in synthesizing it. This is true whether you are synthesizing orchestral instruments, human voices, or vibrating columns of air in pipes, which is essentially what wind instruments are. So with this in mind, let's turn our thoughts to oscillating bodies...

### Of Strings & Pipes

Cast your mind back to the first part of this series (see SOS May '99 or www.sospubs.co.uk/sos/may99/articles/synthsec.htm). If you're old enough, you'll remember that we started all this by considering the nature of a stretched string fixed between two points, and why it prefers to vibrate at certain frequencies that we call 'harmonics'. Figure 1(a) shows such a string, oscillating at its fundamental mode of vibration. Just to remind you, Figures 1(b) and 1(c) show the second and third such modes of vibration.

Without repeating all of the first Synth Secrets article here, we know that the second harmonic oscillates at twice the frequency of the first, that the third oscillates at three times the frequency of the first... and so on. In fact, a perfect stretched string can, in theory (although not in reality), generate the complete harmonic series from 1 to infinity, and the amplitudes of each of them will determine the shape of the audio waveform thus produced. For example, if the amplitude of any harmonic 'n' is 1/n times that of the fundamental, we obtain a sawtooth wave. Alternatively, if the amplitude of any odd-numbered harmonic 'n' is 1/n times that of the fundamental, but all the even harmonics are missing, we obtain a square wave. But what has this got to do with pipes? Well... rather a lot, as we shall see.

Let's now consider the open rigid pipe shown in Figure 2. As the pipe isn't suspended in a vacuum, it has a column of air inside it. It might appear that the air can enter and exit without anything special happening; however, as you know from everyday experience, if you blow across the top of such a pipe, it will generate a pleasant note. You may therefore assume (correctly) that the air inside it is oscillating in such a way as to produce a harmonic series. But how?

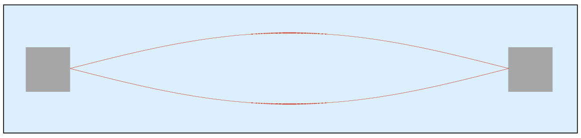


Figure 1(a): The stretched string.

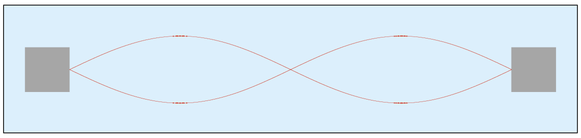


Figure 1(b): The second mode of vibration of a stretched string.

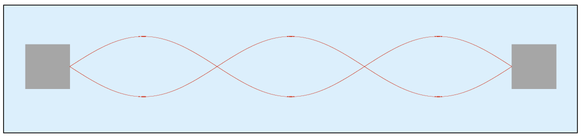


Figure 1(c): The third mode of vibration of a stretched string.

Imagine that you put the tube to your mouth and blow a single, almost instantaneous puff of air into it. In doing so, you create a pulse of higher pressure at one end. The high-pressure pulse passes through the pipe until it reaches the far end, at which point you might expect it to leave the pipe and vanish — but the reality is very different. Far from being released, most of the energy is reflected back into the pipe, almost as if it had bounced off an invisible wall. Likewise, when the reflected pulse reaches the other end, most of the energy is again reflected inwards, and this continues until all the energy is dissipated.

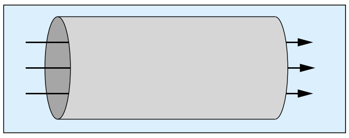


Figure 2: Air moving through a pipe.

It's hard to explain in words, but the reason pipes produce a harmonic series is very similar to the reason why strings do. A string has to be fixed at both ends for harmonic motion to occur; this is called a boundary condition. The pipe has an analogous condition: the pressure of the atmosphere outside the pipe is, well, the pressure of the atmosphere. Therefore, the pressure at the precise ends of the pipe must also be the local atmospheric pressure, or else all the air would either rush out of, or into, the pipe itself.

It's a reasonable guess, therefore, to assume ▶

that — given the right type of blowing — the maximum positive pressure in the pipe will occur at the centre, as will the maximum negative pressure of the reflected pulse. These are the conditions under which a standing wave can occur, as I've tried to make clear in Figure 3(a).

As you can see, this is a very similar diagram to Figure 1(a), representing the stretched string. Having read the first part of Synth Secrets, you shouldn't find it requires a great leap of understanding to realise that Figure 3(a) represents the fundamental frequency of the pipe. What's more, it's easy to see what the wavelength is... the pressure pulse must travel both down and back up the pipe to complete one cycle, so the wavelength of the fundamental is twice the length of the pipe.

So, for example, if the pipe is 0.34 metres in length, and given that the speed of sound is 340m per second, the pulse will travel down the pipe in a thousandth of a second (1mS). It will then pass back up, also in a thousandth of a second, so the period is 2mS, and the frequency of the fundamental is therefore 500Hz. Do these figures look familiar? They should, because they're identical to our example of a resonant body in part 22 of Synth Secrets.

There's another way in which a pipe is very much like a plucked string... provided that the boundary conditions are met, many different modes of vibration are possible within it. Figures 3(b) and 3(c) represent the second and third harmonics of the pipe. Again, the similarity to the stretched string is startling, and like the string, the pipe will support many such modes simultaneously. Indeed, since every harmonic is possible, we could even envisage situations where pipes produce sawtooth waves and many other complex tones.

## Closing The Pipe

Despite all this, we have not reached the point where we can discuss pipes as musical instruments. This is because the oscillations within them have — apart from a brief, hypothetical puff across one end of the pipe — appeared as by magic, with no visible means of generating and sustaining the standing waves I have described. In other words, there is no realistic energising mechanism in this description. So now we must add a means to give energy to the column of air, and to sustain the standing wave within it.

This mechanism is you... or, rather, those people who can play a wind instrument. And the means by which they do so is by blowing into a mouthpiece. But if you are a total novice and try to do this, you will probably be unable to produce a pure note. What's more, you will find that your lips tingle after just a few seconds, and this can be a bit uncomfortable. The reason for is quite unexpected... you may believe that you are blowing a continuous stream of air down the pipe but, in fact, you're producing a stream of pressure pulses. Your lips are, therefore, acting as a valve, creating short pulses of high-pressure air, each followed by a lull before the next. If the timing of these pulses is appropriate to the length of the pipe (and therefore the wavelength, and therefore the fundamental frequency or a harmonic) you will produce a note. If not, a strangled fart is most likely to ensue. Good players control this timing by regulating the tension of their lips and the air pressure in their mouths. Bad players are unable to do this, and we've all heard the results of that.

If you think that this is all straightforward, it isn't. By placing your gob over one end, you're changing something fundamental (no pun intended) about the properties of the pipe. To be precise... you're closing one end. This means that the boundary conditions have changed, and we can no longer trust our previous analysis.

Fortunately, it's not hard to understand the changed response of the pipe. If you look at Figure 4(a), you'll see that the boundary condition at the right-hand end is unchanged. The pipe is still open here, and the pressure differential (compared to the outside atmosphere) must still be zero. However, the closed end (ie. where you blow) is now a region of maximum pressure, and the fundamental waveform now appears as shown in Figure 4(a).

I should warn you that this is a crude way of describing a complex phenomenon, but it nonetheless suggests a very important consequence of blowing into a pipe. Whereas the air only needed to flow down and back up the pipe once to complete a cycle in an open pipe, it needs to complete the round trip twice in a closed pipe. This means that the wavelength of the fundamental is now four times the length of the pipe, rather than double! The musical consequence of this is obvious: the closed pipe produces a note one octave lower than you would otherwise expect. Given our 34cm pipe, the wavelength is now 136cm (instead of 68cm as before), so the fundamental lies at 250Hz rather than 500Hz.

**"When you choose a waveform on a synth, the type of filter and its cutoff frequency, and the amount of filter resonance... what are you really doing? You're determining the harmonic content of the sound."**

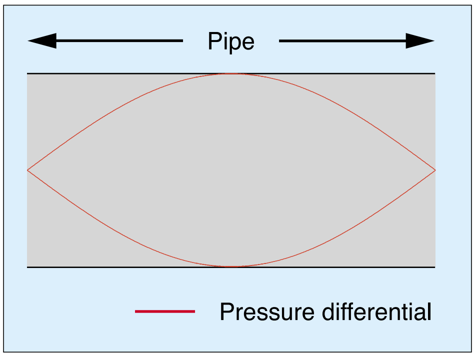


Figure 3(a): The pressure of the initial pulse, and that of the reflected pulse in a simple pipe.

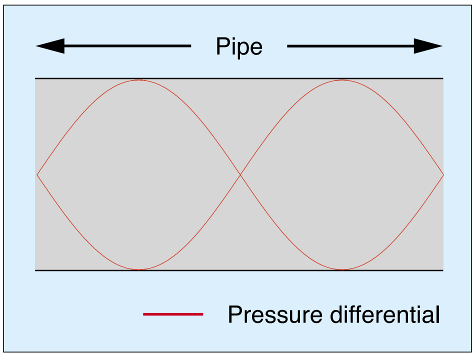
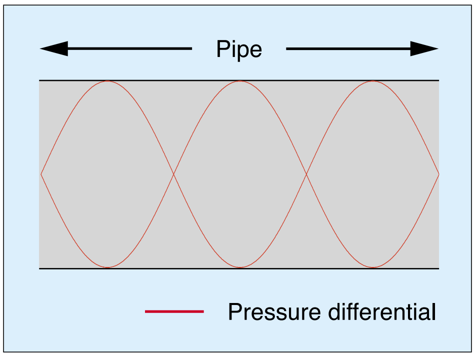


Figure 3(b) (above) and 3(c) (below): The second and third harmonics of an open pipe.



You might ask whether there are any other changes caused by closing one end of the pipe, and you would be wise to do so. Let's look at the second and third modes of vibration, just as we did for the open pipe in figures 3(b) and 3(c).

By studying figures 4(b) and 4(c) you'll see that, in 4(b), one length of the pipe holds three-quarters of a full wavelength. For the next mode, shown in 4(c), one length of the pipe contains 5/4 (ie. one and a quarter) wavelengths. This means that the wavelength at 2/4ths, 4/4ths, 6/4ths (and so on) are missing in a closed pipe. In other words, unlike an open pipe, a closed pipe produces only odd harmonics!

## The Characteristics Of Wind Instruments

OK, that's enough theory for the moment. Let's now ask ourselves how we can use this knowledge to understand and synthesize monophonic wind instruments.

Let's start with the simplest of all such instruments... the recorder. In this, instead of using your lips to excite the air, you blow into the mouthpiece and an open edge creates the pulses that set up the standing waves within the pipe. The consequence of this is that the recorder is, to all intents and purposes, an open pipe. It has a fundamental wavelength of twice the distance from the edge to the end of the bore, and produces all the overtones in the harmonic series (see Figure 5). This suggests that a sawtooth wave or a triangle wave would be suitable for synthesizing the recorder.

Now let's move on to a more complex instrument, the clarinet. Ignoring for the moment the holes along its length and the small 'bell' at its end, this is another cylindrical tube. However, instead of having an edge to cut the air and set up the vibrations, it uses a single reed that acts in the same fashion as the 'lip valve' I described above. Since it is, therefore, a closed pipe, we might expect it to produce a waveform that contains only odd harmonics, and whose lowest note has a fundamental wavelength of four times the length of the instrument. And we would be right. The clarinet has a very distinctive 'hollow' sound which, on a synthesizer, we associate with square waves... the only common synthesizer waveform that contains just odd harmonics.

The difference in harmonic structure between the recorder and the clarinet has an interesting consequence. If you 'over-blow' a recorder (ie. blow harder than is necessary to create the standing wave of the fundamental) you will eventually make it jump up a mode, so that the second harmonic becomes the lowest pitch produced. If the fundamental is, say, C3, the first overblown note will be C4, followed by G4, C5, E5... and so on up the complete harmonic series. In contrast, the first overtone of the clarinet is the third harmonic, and the second is the fifth harmonic, so the equivalent series will be C3, G4, E5... and so on.

Now let's turn our attention to the brass family... instruments such as bugles, cornets, trumpets, trombones, and tubas. These are 'lip-valve' closed pipes, so we might expect them to act and sound like the clarinet, producing odd harmonics only, and having a characteristic 'hollow' timbre. But experience tells us that these instruments are very brash, and that the square wave is quite unsuitable for synthesizing them. Indeed, if ever there was a family of sounds that demanded the use of the sawtooth wave, this is it. So what's gone wrong with our analysis?

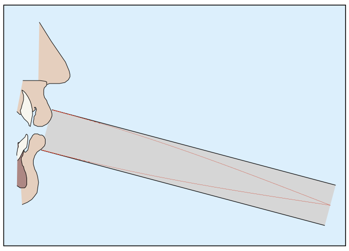


Figure 4(a): The pressure in a blown pipe.

The answer is to be found in the bore of the instrument. The recorder and the clarinet are cylindrical pipes, whereas much of the bore of a brass instrument is conical. Does this make a difference? You bet it does... Apart from the open pipe, there are two shapes of bore that generate the complete harmonic series. One is the cone, and the other is the flare — a geometric shape in which the radius of the bore increases by a constant factor for each doubling of the length of the pipe.

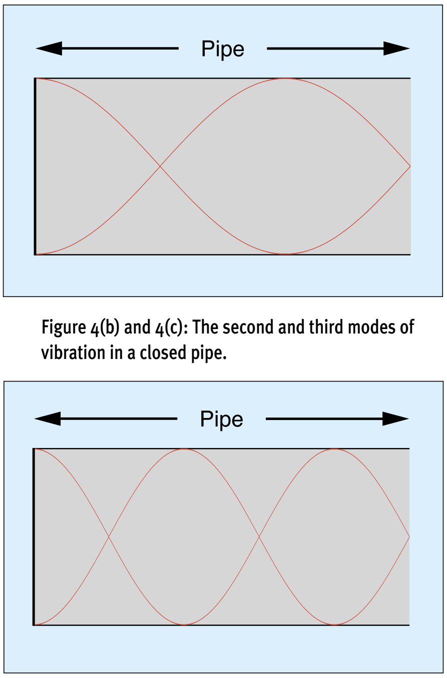


Figure 4(b) and 4(c): The second and third modes of vibration in a closed pipe.

## More About Harmonics

Of the three instruments described, only the clarinet acts like a closed pipe. This suggests that the recorder and the trumpet — both of which produce the full harmonic series — should sound dissimilar to the clarinet, but quite similar to each other. In practice, experience tells us that this is far from true, so how can we further differentiate between them?

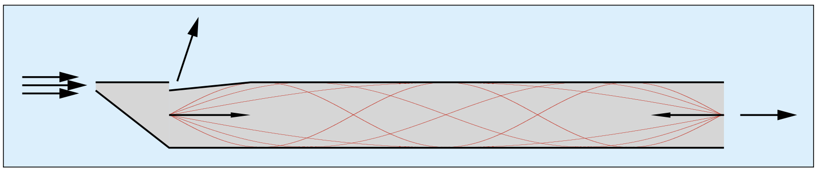
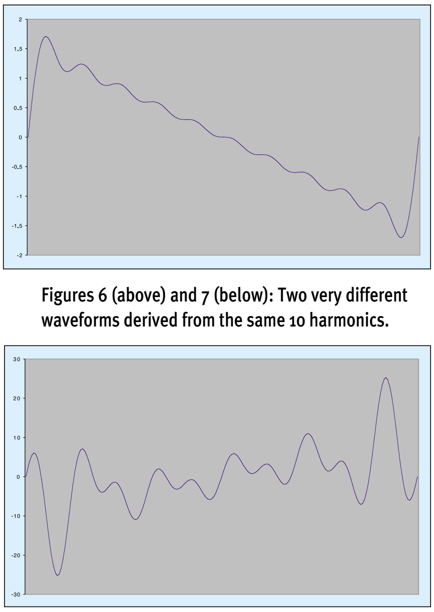


Figure 5: The recorder.

Consider Figures 6 and 7. These depict single cycles of very different waveforms, but both are constructed from the same 10 harmonics. In the first case, I have given them amplitudes that conform to the 1/n relationship that defines a sawtooth wave and, as you can see, the result is very similar to an ideal sawtooth wave. However, in the second example I have chosen a handful of amplitudes at random. The number of harmonics and their frequencies are identical in both Figures... just the relative amplitudes are different. As you might expect, these waveforms not only look very different, but they sound very different too.

So it's not enough to know just the number and type of harmonics



Figures 6 (above) and 7 (below): Two very different waveforms derived from the same 10 harmonics.

present in a sound, we also need to know the relative strengths if we are to analyse and recreate the correct timbre. To illustrate this, I've drawn Figure 8, which shows the relative strengths of the first 20 harmonics of the same note played both quietly and loudly on the same brass instrument. You can learn a lot from this diagram. For example, the fundamental is the dominant harmonic of the quiet note, so you hear this as the perceived note. In contrast, the eighth harmonic is dominant in the over-blown note; you would hear this as a squawk three octaves higher than the fundamental. Also, the quiet note contains just six harmonics, making it 'soft', whereas the loud note includes significant amplitudes of at least 15 harmonics so, quite ignoring the additional loudness of the note, the sound is brighter.

We can extend our analysis still further to ask how the harmonic structures of the notes change, not when we play them louder of softer, but higher or lower in pitch. Analysis (and Figure 9) shows that lower notes appear to have more complex harmonic structures than higher ones. But maybe this isn't so strange. You would expect the lump of resonating metal, wood and air in your hands to have a finite frequency response, with the result that there's less 'room' for the harmonics of higher-pitched fundamentals. Furthermore, much of the equipment used to perform measurements of this type is similarly limited (how high does your microphone go?), so it's a moot point to discuss whether an instrument's harmonics reach 25kHz, 30kHz, 50kHz... or whatever. A typical microphone won't detect these frequencies, the sound system replaying your synthesized sound won't reproduce them, and you wouldn't hear them even if they were there!

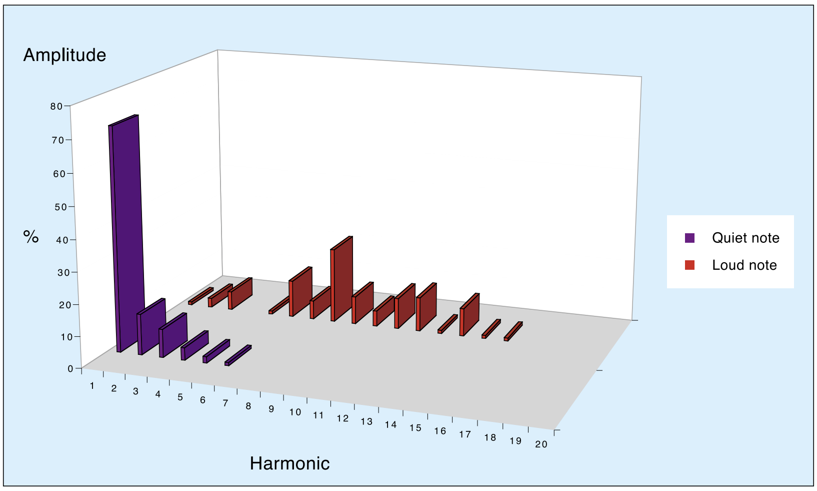


Figure 8: The harmonic structures of the same note played quietly and loudly on the same instrument.

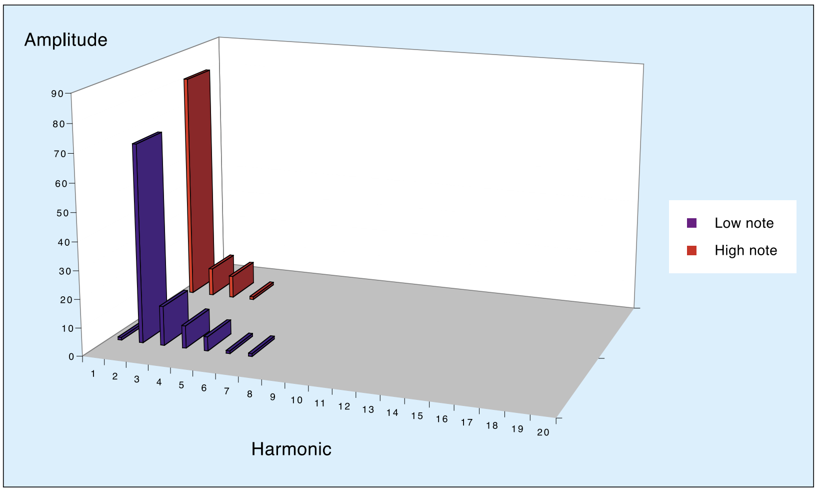


Figure 9: The harmonic structures of a low-pitched note and a high-pitched note played with the same amplitude.

## Other Components Of The Sound

By this point, you must be getting pretty tired of all this talk of harmonic structures. But ask yourself this: when you choose an oscillator waveform on a synth, the type of filter and its cutoff frequency, the amount of filter resonance, and the nature of any contour generators or modulators applied to the cutoff and resonance... what are you really doing? The answer is... you're determining the harmonic content of the sound you're synthesizing. So let's end this month by considering some of the factors we need to consider when we synthesize a trumpet (see Figure 10).

- Firstly, we now know that a trumpet produces a complete harmonic series with significant amplitudes of higher harmonics present. Only one common waveform fits the bill; we must set our oscillators to produce a sawtooth wave.
- Secondly, we know that, as the note gets louder, it contains more harmonics, so we must set up a low-pass filter whose cutoff frequency rises and falls as the loudness of the note increases and decreases, respectively.
- Thirdly, we know that the relative amplitude of the lower harmonics decrease as the note gets louder. This means that we must introduce filter resonance (or some other form of EQ) that emphasises higher harmonics as the loudness of the note increases.
- Fourthly, we know that a high note has fewer harmonics than a low note, so we must set up our filter tracking such that, as the pitch rises, the cutoff frequency rises more slowly, thus tapering the harmonic series.

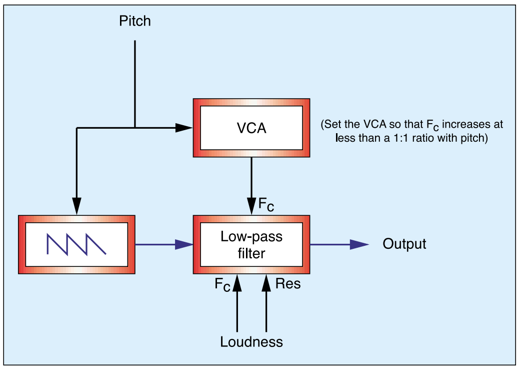

Pitch
VCA
(Set the VCA so that $F_C$ increases at less than a 1:1 ratio with pitch)
$F_C$
Low-pass filter
Output
$F_C$
Res
Loudness

Figure 10: Synthesizing some of the fundamental characteristics of a trumpet.

If we do all of this, we have a chance of creating a tone that is reminiscent of the trumpet we are trying to emulate. Sure, we've yet to consider the transient response at the start of the note, or any changes in amplitude and brightness that occur as we sustain and release the note. Furthermore, we've ignored all forms of modulation and expression, as well as the formants of the real instrument. But this is not a problem. We can — and will — do these things...

Next month, we'll look at other attributes of brass and woodwind sounds, and put together a few basic — and not so basic — analogue synth patches to emulate them. Until then... SOS

# synth secrets

## PART 25: SYNTHESIZING BRASS INSTRUMENTS

**Gordon Reid** builds on the acoustic theory of wind and brass instruments he introduced last month, and explains how to produce a convincing analogue trumpet sound.

Before going any further, I owe you an apology. At the end of Part 23 of Synth Secrets (see *SOS* March 2001) I said that we would crack on with some genuine synthesis at the expense of some of the acoustic or electronic theory. I then gave you Part 24, which delved into a great deal of acoustics, and offered little in the way of synthesis. For this I must apologise, but the further I probed into the nature of pipes, the more I realised that it would take a couple of chapters to tell the whole story of synthesizing brass and woodwind instruments. Indeed, university textbooks offer hundreds of pages about pipes and the instruments created from them. Nevertheless, I have tried to condense the topic down somewhat. So this month, I offer you a practical approach to synthesizing brass.

### Beyond The Initial Waveform

Last month I finished with a quick example: a look at how to create the basic elements of a trumpet-like waveform by subtractive synthesis. But as I pointed out in my final paragraph, that static approach ignored all consideration of how a real trumpet note evolves and changes over time, and that's what we'll examine more closely now.

Ask yourself this: when you hear two instruments play a note of identical loudness and pitch, how do you differentiate between them? Indeed, let's make the question even more difficult: when you hear two instruments that produce the same harmonic series play a note of identical loudness and pitch, how do you differentiate between them? For example, how can you tell the difference between a recorder playing a C, and a trumpet playing the same note? Both produce complete harmonic series (although for different reasons — as explained last month, the recorder does so because it's an open pipe, while the trumpet does so because it's not cylindrical along all its length) yet they sound very different, both when playing individual notes, and in performance.

The reasons for this lie in the many other factors that determine the sound of a given type of instrument (or, for that matter, of individual instruments within a family). These factors include the following changes that occur during the course of a note:

- the loudness contour of the pitched note;
- the tonal contour of the pitched note;
- the pitch contour of the pitched note;
- the instrument's formants;
- the attributes of any noise present in the signal.

As you can see, this is a far from trivial list. So let's return to the trumpet example, break the sound down into its individual parts, and see how we can use the sections of a modular analogue synthesizer to recreate them.

### The Amplitude Response

By definition, the initial (or 'transient') response of a brass instrument is that section of the sound that occurs at the beginning, before the note settles down into its 'steady state'. Although some of the attributes of the transient are pre-determined by the nature of the instrument, the player can also influence its nature by adopting different tonguing and blowing techniques. These techniques allow the player to alter the pressure that initiates the standing wave in the tube (refer back to last month if this talk of standing waves is confusing you). This means that the amplitude response of the transient can be gentle, rapid, or even (if the air is expelled violently) 'plosive'.

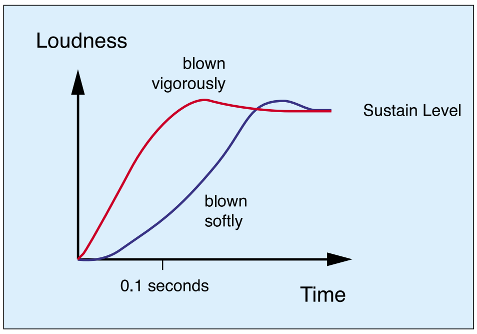


Figure 1: The transient amplitude response of a pipe blown softly or vigorously.

If you measure the amplitude changes of the transients in softly and vigorously blown pipes, you get similar results (see Figure 1 above). What's more, the response describes a shape you should recognise. Although the durations of the stages differ, you can approximate these curves using the Attack, Decay and Sustain stages of a conventional four-stage ADSR synth contour generator.

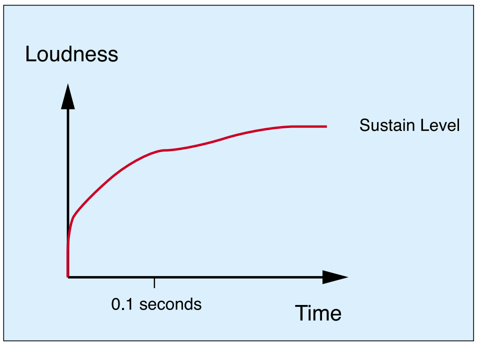


Figure 2: The summed plosive transient.

Of course, this is no accident. Pioneers such as ▶

▶ Bob Moog and Alan R Pearlman (of ARP) chose the ADSR contour for good reasons, and it's something that you'll find on nearly every analogue synthesizer. So now we know what we're going to use for the first part of our output (audio) VCA contour!

Now let's turn to the plosive case, which occurs when players use their tongues to speak a 'T' or 'D' while blowing the instrument. This is more complicated than the other examples, because the transient amplitude does not follow an ADS contour. Instead, it exhibits an almost instantaneous Attack at the instant you blow the note, followed by a slower second Attack before reaching the Sustain Level (see Figure 2, previous page). This means that you need a different type of contour called an A(AL)A2S, which stands for Attack, (Attack Level), Attack2, Sustain. Unfortunately, although you can find such an envelope on certain digital synthesizers, it's not available on analogue synths. But don't despair... physics comes to your aid here. If you create an ADS contour with, theoretically, an instantaneous Attack, the limitations of the electronic circuits 'round off' the transient, and you get a contour similar to that generated by the real instrument (see Figure 3, right).

Photo courtesy of the EMIS Synth Museum

Despite the complexity of a real brass sound, careful use of subtractive synths like the Oberheim OBX can produce some fine-sounding analogue brass sounds.

We're now in a position to define one of the stages in the synthesis chain of our brass sound, as shown in Figure 4 (right). Note the extra module in the block diagram. This is the loudness sensor which, on most synthesizers, will be a keyboard velocity sensor. You route this so that, as you hit a key harder (thus simulating the increased air pressure as the brass player blows into the mouthpiece harder and harder) the transient gets faster and faster. In practical terms, this means that the Attack stage gets shorter as the velocity increases. Unfortunately, most vintage synths do not have this facility so, to emulate this effect, you may have to tweak the Attack knob or slider by hand as you play.

Now that the amplitude of the sound has reached its steady state, you have to consider what happens to it until the end of the note. Simplifying matters somewhat, you find that there are three other main factors to consider.

The first of these is tremolo — delayed or otherwise — that the player may add by modulating the air pressure once the note has reached the steady state. We can synthesize this by applying an LFO to the gain of the VCA. The second is any swell or *diminuendo* the player may apply during the course of the note, and it's easy to generate this if the synthesizer has a suitable controller or a contour generator with more than four stages. The final factor is the time it takes for the note to die away when the player stops blowing. This is the simplest of all... it's the amplitude contour's Release time. On a brass instrument, this is very short. You can generate all three of these factors as shown in Figure 5 (below).

If I now put Figures 4 and 5 together, I get the envelope shown in Figure 6(a), below. This is a typical 'swell brass' envelope, and it is easily produced by a synth with a five-stage amplitude contour and an LFO that can be routed to the audio signal VCA.

Unfortunately, few analogue synthesizers offer five-stage contour generators, and most of us make do with the traditional four-stage ADSR generator. This means that the contour in Figure 6(a) is out of the question, and we must limit ourselves to the amplitude envelope shown in Figure 6(b). But don't despair... some of the greatest synthesized brass sounds ever created emanated from synths with ADSRs, such as the ARP Odyssey, Sequential Prophet 5, Oberheim OBX, and Moog Memorymoog, to name just four.

## The Tone Contour

Now let's consider the brightness of the sound. Last month, I looked at the steady-state spectrum of the waveform as produced at different pitches, but let's quickly recap. You should recall that louder notes have more harmonics than quieter ones. Figure 7 shows how we can achieve this using the pitch CV and a loudness sensor to affect the cutoff frequency of a low-pass filter. Furthermore, you know that the loudest ▶

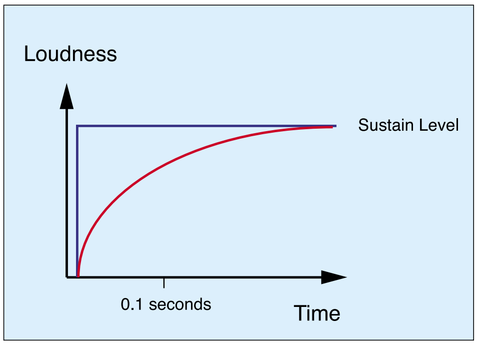


Figure 3: The real response of an ADS circuit.

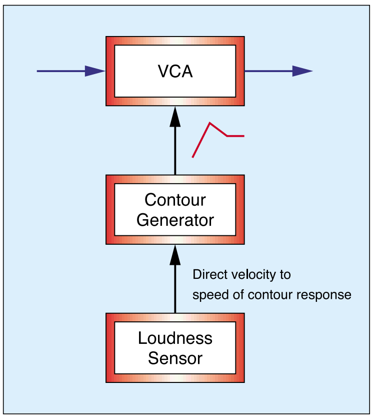


Figure 4: Creating the initial (ADS) amplitude contour of the pipe-based instrument.

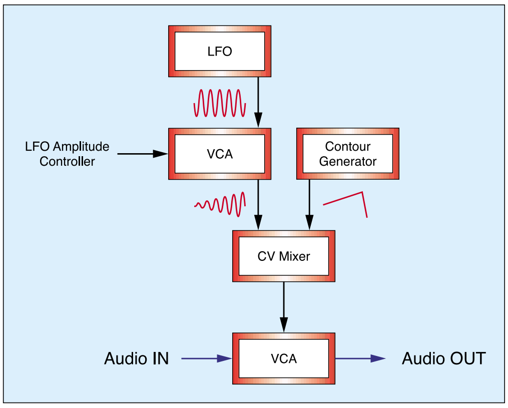


Figure 5: Generating the Sustain and Release stages of the amplitude envelope.

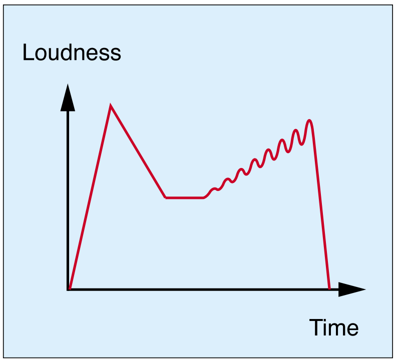


Figure 6(a): The loudness envelope of a note defined by a five-stage contour plus LFO.

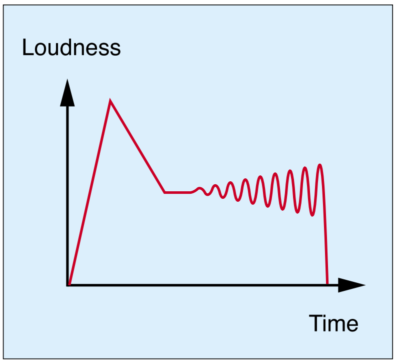


Figure 6(b): The loudness envelope of a note defined by a four-stage contour plus LFO.

harmonic need not be the fundamental (as in overblown notes). Again, Figure 7 provides a solution: by making the resonance proportional to the loudness, you ensure that high harmonics are accentuated to a greater and greater degree as we play louder and louder. But neither last month's analysis nor Figure 7 tells you what happens at the start of the note...

You might think it obvious that the player's tonguing and blowing techniques would change the loudness contour of the transient. However, you might be more surprised to hear that they also alter the mix of harmonics within the transient, thus making the timbre more or less aggressive. Vigorous tonguing (no giggling at the back, please) can even make high harmonics the loudest, thus pitching the transient an octave (or more) higher than the fundamental — see Figures 8, 9 and 10.

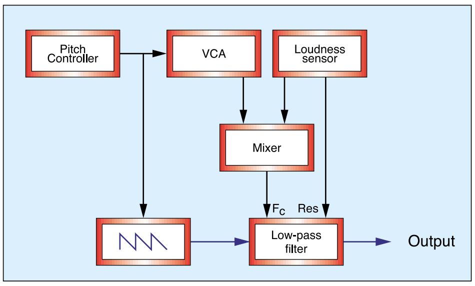


Figure 7: How the loudness can control the harmonic characteristics.

If you study Figure 8, you'll see that the loudness of the third harmonic is still increasing after the fundamental has attained its peak level. In Figure 9, all the harmonics appear to attain their steady-state amplitudes at the same time. In fact, neither description is quite right. Ignoring the plosive case (which would require an instalment of Synth Secrets to itself) it's a reasonable approximation to say that the harmonics beneath the instrument's natural cutoff frequency (for any given note) reach their sustain levels together, and more quickly than the harmonics above the cutoff point (see Figure 11).

This might seem to be in conflict with the amplitude graphs in Figure 1, but it isn't. By and large, the amplitudes of the higher harmonics are low compared to those of the fundamental and lower harmonics, so the slow development of the higher harmonics has little effect on the amplitude envelope. Therefore the red line in Figure 11 (the summed amplitude of the lower harmonics) looks almost identical to the red line in Figure 1, which shows the overall transient response.

However... while the tardiness of the higher harmonics has little affect on the development of the amplitude, it has a huge effect on the tone of the sound. Indeed, some researchers believe that the differences in the development rates of the harmonics are the most important audible clue you have as to the identity of an instrument when you hear it.

In theory, you can recreate a response similar to that shown in Figure 11 using a simple low-pass filter and an associated AR contour generator to create the filter profile shown in Figure 12. But if you try this, you'll find that your brass sounds are very unconvincing. To a large extent, this is because we have overlooked the 'parp' of the initial overblown pulse of air released by the player to initiate the note. To recreate this, you need a four-stage ADSR contour generator, as shown in Figure 13. For realism, you also need to add velocity sensitivity to the patch, and you do so by hooking up a velocity sensor to control the amount of ADSR contour applied to the VCF cutoff frequency (see Figure 14). As you can see, this diagram has an extra module that I've not used before: a pressure (or 'aftertouch') sensor. Hooking this up to the gain of the VCA controlling the envelope amount allows you to vary the brightness during the sustained part of the note, just as a 'real' brass player would do.

## Pitch Modulation — Growl & Vibrato

This still isn't the end of the story. All brass instruments require a 'settling time' at the start of the note. This is because it takes a finite amount of time for the standing wave to reach its steady state. For a note of, say, 256Hz (middle C), this 'settling' takes about a dozen cycles. This means that there's a period of pitch instability lasting about 50mS: a duration of the same order as the time it takes for all the harmonics to reach their steady-state amplitudes (this explains why all horn players sometimes fluff the initial pitches of notes... to a large extent it's the instrument's fault, not that of the players!).

Now, you might expect to apply some sort of modulation to the oscillator to emulate this, but there's a good reason not to do so. Any form of periodic or even quasi-periodic modulation applied to the frequency of the oscillator will result in frequency modulation (FM), and therefore lead to the production of side-bands (see part 12 of Synth Secrets, back in *SOS* April 2000, or check out www.sospubs.co.uk/sos/apr00/articles/synthsecrets.htm). This would destroy the timbre of the brass patch, and take it off in a new direction entirely. Instead, I apply a modulator to the low-pass filter in a way that achieves the desired effect.

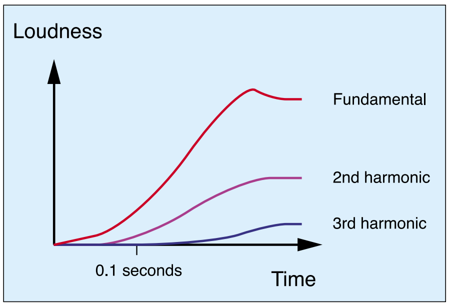


Figure 8: The transient response of the first three harmonics of a pipe blown softly.

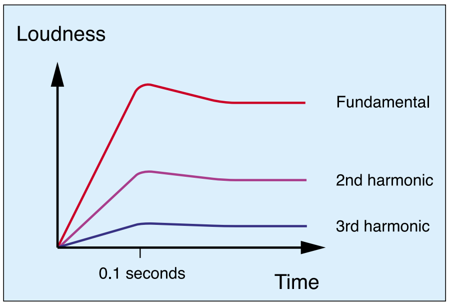


Figure 9: The transient response of the first three harmonics of a pipe blown abruptly.

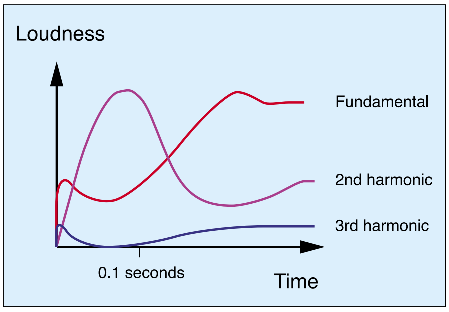


Figure 10: The transient response of the first three harmonics of a pipe blown plosively.

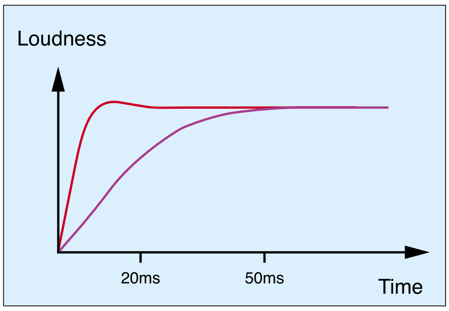


Figure 11: Higher harmonics take longer to 'speak' than lower ones.

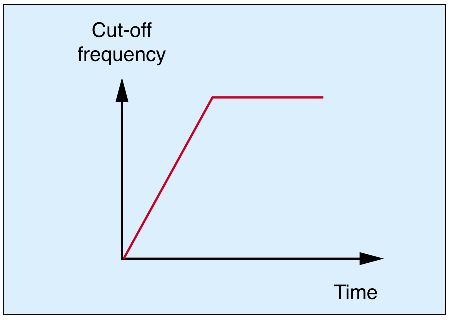


Figure 12: The brightness contour.

Over the years, I've found that a triangle wave is an acceptable source for this modulation, and a frequency in the region of 80Hz does the trick nicely (this, by the way, is one of the reasons why I point out that a maximum LFO frequency of, say, 25Hz is inadequate when reviewing synths). Of course, you don't want the resulting growl to last for the duration of the note; it should last as long as the instability in the acoustic instrument we're synthesizing. Therefore, we patch the modulation into the filter through a VCA whose gain is controlled by an AD contour generator, as shown in Figure 15.

> **"Few analogue synths offer five-stage contour generators, and most of us make do with the traditional four-stage ADSR generator. But don't despair... some of the greatest synthesized brass sounds ever created emanated from synths with ADSRs."**

Again, I've patched a pressure sensor to the gain control input of the VCA controlling the amount of growl. I discovered how useful this could be when I bought a touch-sensitive ARP Pro-Soloist, which allows you to re-introduce the growl using aftertouch. This is a fantastic way to add expression, imitating those instances when a brass player over-blows the instrument. If you don't have aftertouch sensitivity on your synth (or if you're not using a keyboard synth) you can achieve the same result using other controllers such as joysticks and CV pedals.

There's one more modulation to introduce before the emulated brass sound can be considered finished. This is vibrato, which players introduce and control by adjusting the tension of their lips once the note has reached its steady state. Since vibrato does not occur during the transient stage of the note, you can't simply apply an LFO to the oscillator. Delayed vibrato is what is required, and it's usually implemented as an AR ramp controlling the amount of modulation. Figure 16, right, shows the patch that creates this. I find that modulating frequencies in the region of 5Hz sound the most realistic, and that the amplitude of the modulation must be very low, otherwise the timbre will sound electronic.

## Noise

In addition to everything above, noise forms part of the signal in any pipe-based instrument. This is because some of the air within a real instrument will become turbulent, unlike the idealised cases I discussed last month. Scientists refer to this noise as 'aerodynamic' or 'turbulent' noise.

If the pipe did not confine this turbulence, it would be largely 'white' within the limits of its bandwidth. However, the formants of the instrument shape the noise, as does the player's technique. For some instruments — pan-pipes are a perfect example — the noise content is a large part of the sound's appeal, but for most orchestral instruments this is not the case, and it should be almost undetectable in normal playing. What it adds, however, is a low-level, tuned undertone to the sound — sometimes harmonically related to the note, but often not. If we add noise in this way (see Figure 17) it can add a great deal of realism to the synthesized sound.

## Putting It All Together

Now for the exciting bit. Given a large enough and suitably equipped modular synthesizer, there's no reason why we shouldn't create the entire patch described above. Being realistic, we must make some compromises, using common elements in many places within the patch. Figure 18, on the last page of this article, shows the result. I think that I've incorporated everything from the preceding pages, but if I've missed anything, please don't bother to tell me. You have no idea how long it took me to figure out a way to draw the diagram sensibly...

By the way, if you think that Figure 18 looks like the block diagram from a synthesizer service manual, you would be right, because it includes many of the modules from a single-oscillator synth.

## Conclusion

This chapter of Synth Secrets may seem very detailed, but, in reality, an undergraduate student of acoustics could pick many holes in it. This is because, while I have attempted to explain the most important elements of brass instruments, and describe them from the perspective of a synthesist, the truth is *still* more complex than presented here. For example, we've ignored any conventional pitch

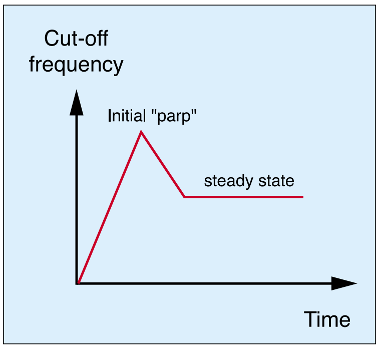


Figure 13: A more realistic transient brightness contour.

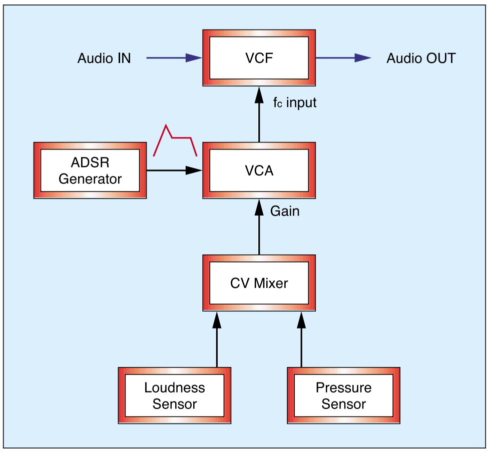


Figure 14: A simple patch to control the brightness of the note.

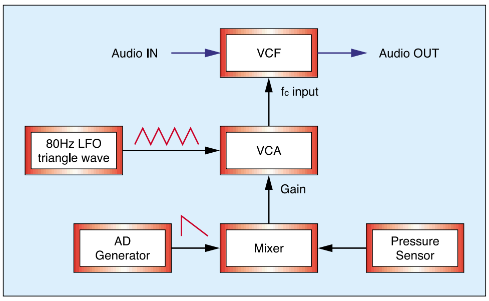


Figure 15: Adding initial 'growl' to the sound.

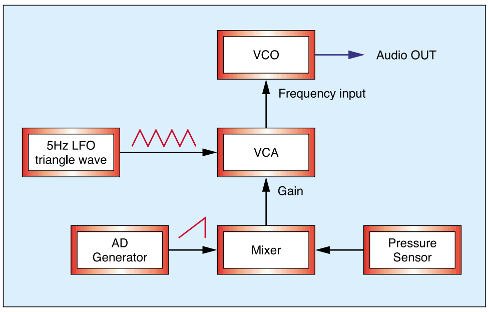


Figure 16: Adding delayed vibrato to the patch.

envelope that the player might wish to create (including portamento effects) and the effects of formants on the timbre of the tonal elements of the sound. I have also ignored the phases of the harmonics which, due to the complexities of brass instruments, are not all in phase with one another. Furthermore, the amplitude of individual harmonics may change during the course of the note, and in ways that we cannot easily emulate using low-pass filters and amplifiers. Even the frequencies of the harmonics (which, at this point, we should strictly call 'partials') are not constant, and can vary between the transient and the steady state. Indeed, for reasons we need not dwell on here, the partials are not, strictly speaking, harmonics at all. Their frequencies are stretched out (sharpened) as the harmonic number increases.

When you look at it like this, you have to conclude that subtractive synthesis is not an ideal way to recreate brass sounds. In theory, microtonal additive synthesis would be a far better way to go about it. Indeed, additive synths such as the Kawai K5 and K5000 are superb at recreating brass-type sounds (see *SOS* January 1997 or www.sospubs.co.uk/sos/1997_articles/jan97/kawaik5000w.html). Nevertheless, if you're careful about it, subtractive synthesis can make more than a passable stab (pun intended) at producing the sounds of brass instruments, and even this simplified analysis should help you to create better patches and better performances.

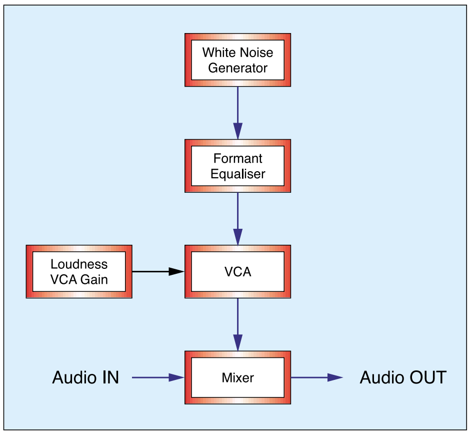


Figure 17: Adding noise to the tonal signal.

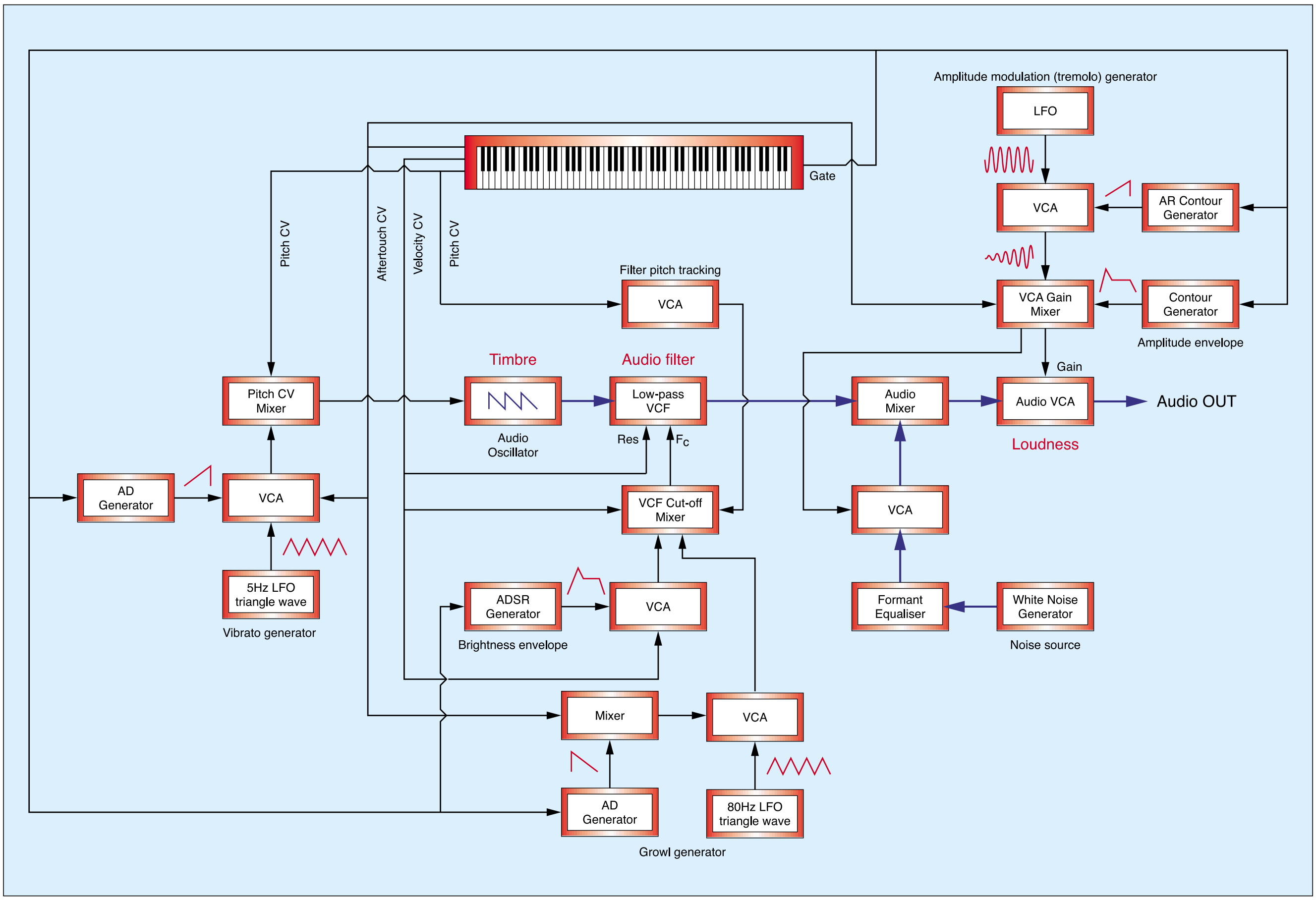
Photo: Mark Ewing

Brass sounds are also one of the many strengths of additive synths such as the Kawai K5000. For more on additive synthesis, check out Paul Wiffen's article on the subject in *SOS* October 1997, and his review of the Kawai K5000W in January the same year.

## Epilogue

So... what do you do if want to put some of this theory into practice on your synth, but don't possess a fraction of the elements needed to recreate Figure 18 accurately in your studio? Don't despair, because next month, we'll look at a number of common monophonic synthesizers, and see how we can recreate the essence of that diagram on more limited equipment. Until then...

Figure 18: Finally... the brass patch!



# synth secrets

I promised last time that we'd look next at how the brass synthesis theory I've been explaining over the last two months translates into real sounds on simple subtractive synthesizers. I'm going to start by taking as an example a single, rather basic monosynth, one with very few controls, very few control routings, and just a single signal path. This may not sound very encouraging, until I tell you that the synth is the Minimoog: very simple, very basic, and yet glorious. Since a decent treatment of brass sounds on the Minimoog takes up more than enough space for one instalment of this series, I'll be continuing next month with the Roland SH101 and the ARP Axxe.

Let's start by taking a peek at Figure 1 (see below), which represents the complete synthesis structure of the Minimoog, controls and routing. At first glance, it may look a little daunting, but you'll soon notice how limiting it is... perhaps the most limited of any multi-oscillator monosynth ever produced. If you want to prove this to yourself, try to draw the equivalent block diagram for the contemporaneous ARP Odyssey or ARP 2600!

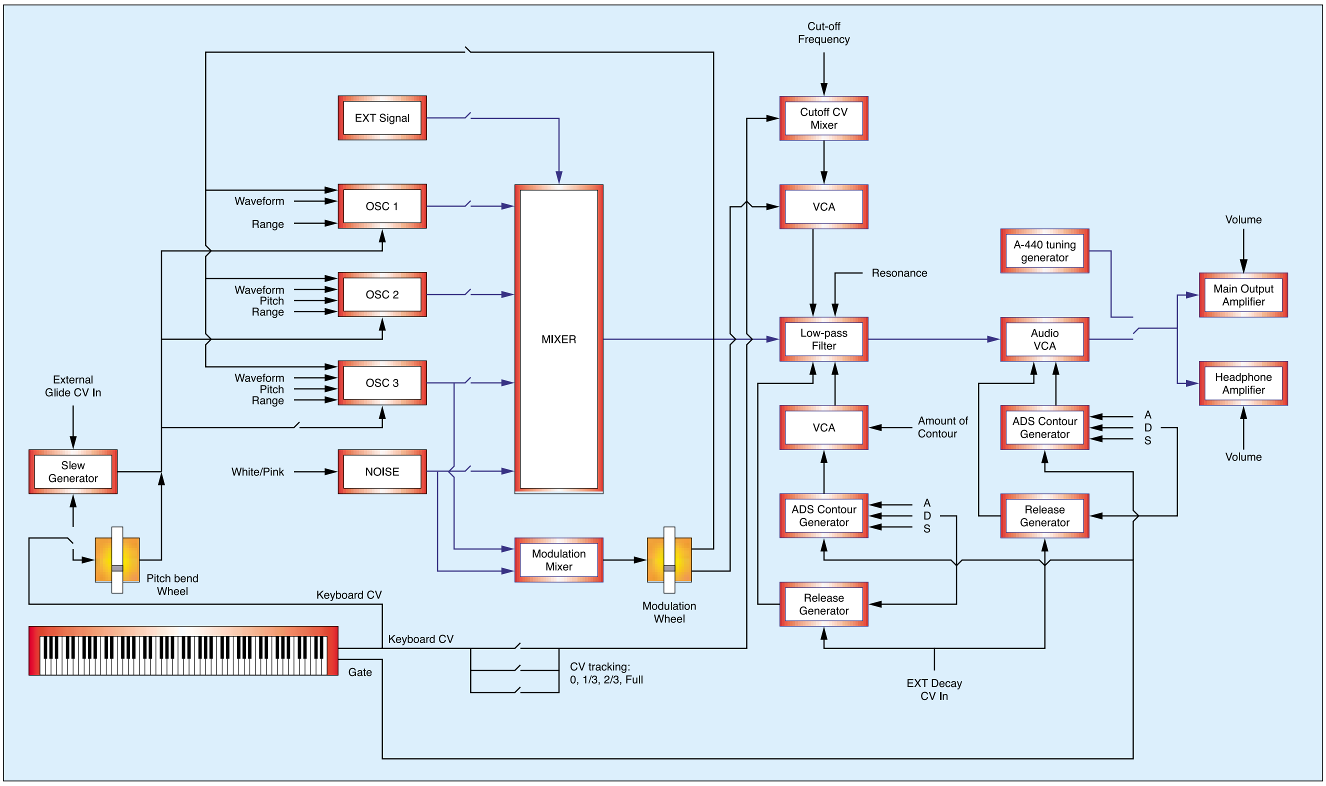
## PART 26: BRASS SYNTHESIS ON A MINIMOOG

Last month we looked at how analogue modules can reproduce the sound of a real trumpet. All very well if you own a wall-sized modular system — but what if your means are more limited? **Gordon Reid** adapts theory to practice with a Minimoog.

Photo: Richard Ecclestone

Now, if you compare Figure 1 to the theoretical brass patch shown at the end of last month's Synth Secrets (shown again in Figure 2, on the right above ), you'll see that the idealised patch requires numerous modules and CV routings unavailable on the Minimoog. This, in turn, suggests that the Moog is incapable of creating a good brass patch. However, experience tells us that the Minimoog is one of the

Figure 1: The structure of the Minimoog.



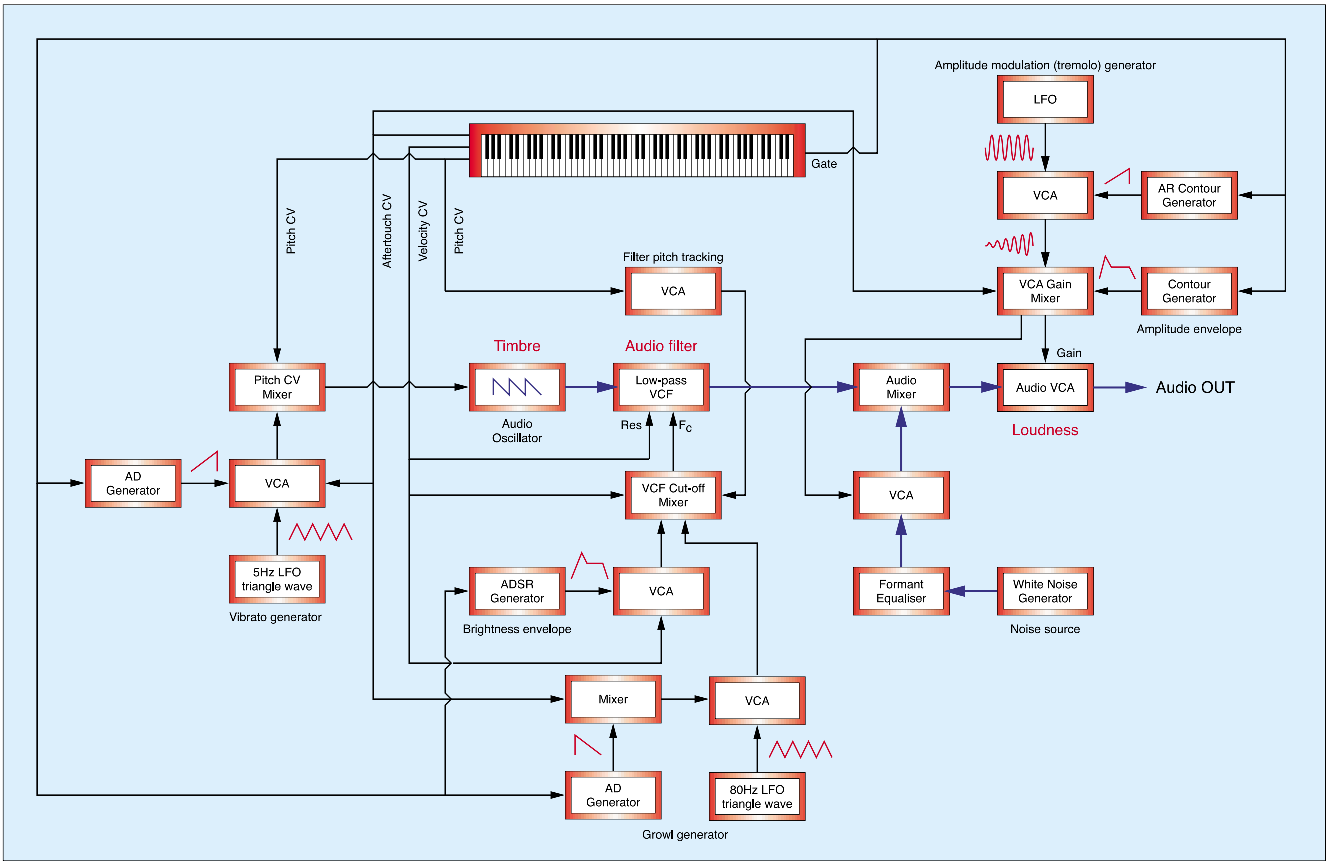


Figure 2: The block diagram for the brass patch in last month's Synth Secrets.

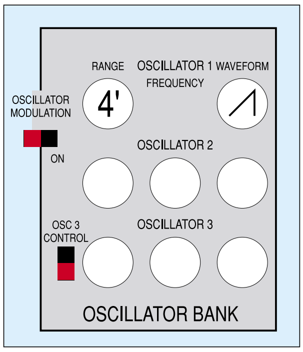


Figure 3: The Moog's Oscillator panel.

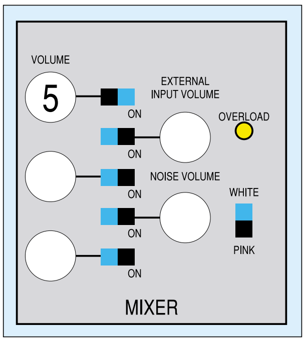


Figure 4: The Mixer section.

best brass synths in the business, so let's find out how!

## The Source Waveform

If you cast your mind back two months, you'll remember that brass instruments generate a complete harmonic series. Since you know that the sawtooth is the only common subtractive synth waveform that does likewise, it should be no surprise that the starting point for your brass patch is a single oscillator producing a sawtooth wave (see Figure 3, left).

You might ask why I don't use all three of the oscillators that the Minimoog provides. Surely this would create a richer sound, and provide more flexibility regarding the precise timbre? There are two reasons for not doing this; one acoustic, and the other practical. The acoustic reason is simple. The interaction of two or more oscillators — which inevitably on an analogue instrument like the Minimoog will detune and drift with respect to one another — is quite unrepresentative of the original instrument. The practical reason is also obvious; the Minimoog has no dedicated low-frequency oscillator (LFO), so we need to reserve Osc3, the Minimoog's third oscillator, for modulation duties (more on this later).

Returning to the Minimoog's Oscillator Bank, as shown in Figure 3 above, you can see that Osc1 is producing the necessary sawtooth wave. You'll also see that I have selected an octave range of 4' for this patch — one octave higher than a piano playing the same note on the keyboard. This is because trumpets, cornets, alto saxophones and soprano saxophones produce high-pitched notes relative to other brass instruments such as tubas, horns and trombones. As for the red switches in the Oscillator Bank, these are both set to Off, which tells you that there will be no oscillator modulation (vibrato) and that Osc3 will not track the keyboard CV.

Moving on, the Minimoog's Mixer section allows you to select which of the oscillators contributes to the audio signal path. It also allows you to add noise and external signals into the mix (see Figure 4, left). As you can see, only Osc1 is set to On, and its loudness is set to five on a scale of zero to 10. This is because the Minimoog's oscillators are capable of overdriving its filter input at higher levels. The mild distortion thus generated is desirable for some sounds, but not on this occasion.

## Shaping The Waveform: Loudness

To filter and shape the sawtooth waveform, I use two modules: the Minimoog's low-pass Filter section, and its audio amplifier, called the Loudness Contour. Traditionally, because it lies next in the signal path, I would now consider the filter. However, for reasons of clarity (and also consistency with last month's analysis), I'm going to start with the loudness contour.

You may recall from last month that you can use a five-stage contour generator and an LFO with an associated contour generator and VCA to create a good approximation of a brass instrument's changes in loudness. Figure 5 shows the loudness envelope thus defined. I also discussed the need for some form of loudness sensor, such as keyboard velocity sensitivity, to allow expression to be added.

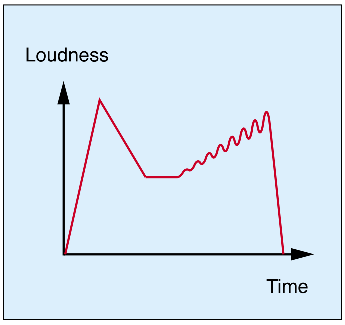


Figure 5: The loudness envelope of a note defined by a five-stage contour plus LFO.

▶ Unfortunately, the Minimoog does not have a five-stage loudness contour, nor does it offer any form of loudness sensor, nor does it have an LFO that can add tremolo! Before describing the best that the Minimoog *can* do, I'm going to jump down to the control panel immediately to the left of the keyboard to determine whether the Decay switch is on or off (see Figure 6, right).

The reason for this is obvious if you know the Minimoog. Far from offering the desired five-stage contours, the Minimoog has only three-stage Attack/Decay/Sustain controls in its contour generators; it cannot even produce a four-stage ADSR contour, as found on most other synths. The best that it can do, when the Decay switch is set to On, is re-apply the Decay time as a Release stage when you lift your finger from a key. Because I know that a real brass sound ends very rapidly once you stop blowing the instrument, I want the synthesized sound to do likewise, so I set the Decay switch to Off.

The only two important parameters in the Loudness Contour section (see Figure 7, above right) are Attack, which should be set to 100 milliseconds, and Sustain Level, which should be set to 10 on a scale of zero to 10. Because the Sustain is at its maximum level, the Decay Time is irrelevant, and the Release Time, as discussed in the previous paragraph, is effectively instantaneous, no matter what the setting for the Decay knob may be. The resulting contour is as shown in Figure 8 (above right). Yikes! It's not much like Figure 5 — but it will have to do.

## Shaping The Waveform: Tone

At this point, you may be wondering how on earth the Minimoog can produce a passable imitation of a brass instrument. If it can, surely it must be because the filter section can produce a close approximation to the ideal? Well, let's see...

You'll recall from the last two months that louder notes have more harmonics than quieter ones. Furthermore, you know that louder notes have higher proportions of higher harmonics than do quieter notes. But there's nothing we can do about this on the Minimoog, because it has no performance controls — velocity or pressure sensitivity, for example — to allow you to introduce this sort of expression. Indeed, there's *no* way to make the intensity of your playing affect the harmonic content of the sound. Fortunately, there are four things that can be done to approximate the tone of a brass instrument.

### ▪ FILTER CONTOUR

We can use the Minimoog's Filter contour generator to allow the higher harmonics to enter the sound in a reasonably realistic fashion. Figure 9 (below) shows the Minimoog's Filter section set up for a brass sound. As you can see, the cutoff frequency is set to -5 (on a scale of -5 to +5) so this is equivalent to 0 percent on most other synths. In other words, the low-pass filter is completely closed unless modulated by some external device. At the same time, the Amount Of Contour control is set to 6.5 (on a scale of 0 to 10) so the associated Contour Generator will sweep the cutoff frequency when you press a note. Remembering that the Decay switch in the performance control panel is set to Off, you can say that the filter contour has an Attack of 600 milliseconds, a Decay of 800mS, a Sustain Level of 5, and an instantaneous Release. I've drawn the resulting contour in Figure 10 below. It's exactly what we want, as shown in Figure 13 last month.

### ▪ RESONANCE

You will see in Figure 9 that the Resonance is set to 2 (out of 10). This suggests that there is a slight bump in the filter cutoff profile, as shown in Figure 11 on the next page. Again, this is very close to the response that the theory and measurement of real brass instruments says we require.

### ▪ FILTER TRACKING

In part 24 of this series, I discussed how the harmonic content of a brass note changes with pitch and concluded that, to reproduce a brass instrument accurately, the filter cutoff frequency must track the keyboard CV at slightly less than a 1:1 ratio. For example, if one note is an octave higher than another (ie. the frequency doubles), the filter should open by a factor slightly less than two... say, to 190 percent of its previous value.

The Minimoog does not offer a variable filter tracking, but instead has four options, selected using the Keyboard Control 1 and Keyboard Control 2 switches in the Filter section. If both switches are set to Off, the filter cutoff frequency does not track the keyboard. If switch 1 alone is on, the filter tracks at 33.3 percent of the keyboard CV. If switch 2 alone is on, the filter tracks at 66.7 percent of the keyboard CV. Finally, if both switches are on, the filter cutoff frequency follows the keyboard CV at exactly 100 percent. The closest approximation to the theoretical ideal is 100 percent, so I have set both switches to On. This means that the resonant 'hump' in the filter profile always lies in the same position relative to the note being played, and that is — by and large — as it should be.

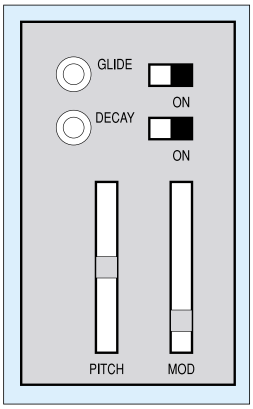


Figure 6: The Minimoog's performance control panel.

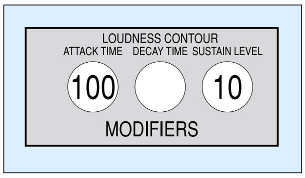


Figure 7: The Loudness Contour settings.

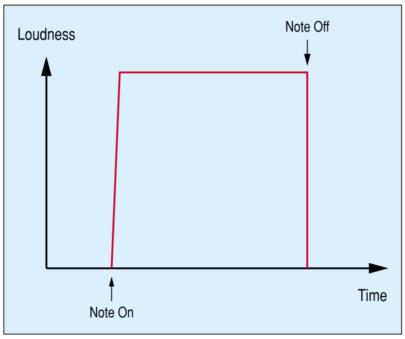


Figure 8: The loudness contour.

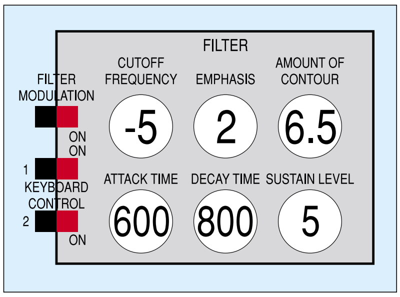


Figure 9: The Minimoog filter section.

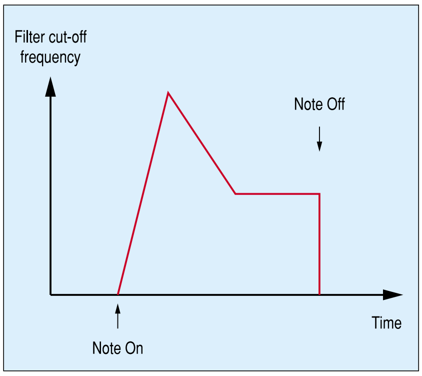


Figure 10: The filter cutoff frequency contour.

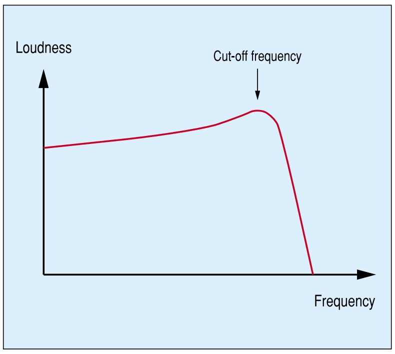


Figure 11: The filter's resonant cutoff profile.

▪ FILTER MODULATION

Last month, I described how a 'settling time' is required at the start of every note played on a brass instrument, and showed how a rapid modulation applied to the filter cutoff frequency could imitate this. I also used a contour generator and a VCA to ensure that this 'rasp' lasted just a fraction of a second.

On the Minimoog, Osc3 can provide the necessary signal to modulate the filter. The bad news is that there is no electronic way to contour its output; it looks as though the filter is either modulated, or not. But don't give up yet...

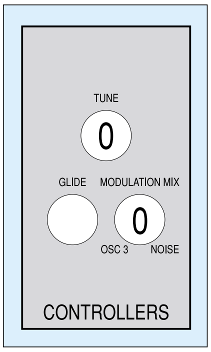


Figure 12: Defining the modulating signal.

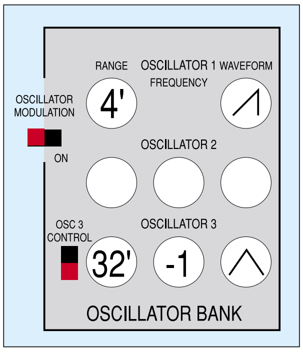


Figure 13: Using Osc3 as an audio-rate LFO.

The filter may be modulated by the output from the Controllers section (shown in Figure 12, above) if the Filter Modulation switch (seen in the Filter section back in Figure 9) is on. The important knob in Figure 12 is the one marked Modulation Mix, which determines whether the modulating signal comprises Osc3's output, the output from the noise generator, or a mixture of the two. I have set the control to 0, giving us just the output of Osc3.

I now need to revisit the Oscillator Bank to set up Osc3 for modulation duties (see Figure 13, above). There are four controls to consider. Firstly, the Osc3 Control switch is off, so the frequency of the oscillator does not track the keyboard CV. This means that the modulation is consistent, no matter what notes we play on the keyboard. Next, the 32' frequency range and the fine-tuning setting of -1 define the frequency I want to use in order to achieve the desired effect. Finally (as discussed last month) the triangle waveform is the one that best allows us to imitate the rasp of a brass instrument.

So far, so good... but I don't want the 'rasp' modulation to last for the entire duration of the note, or it will sound very unnatural. So how do I overcome the lack of a contour generator and VCA to control this?

The answer lies in the performance controls to the left of the keyboard. If you return to Figure 6, you'll see that there's a control wheel labelled 'Mod'. This allows you manually to control the level of the modulating signal. I've shown the architecture of this in Figure 14 on the next page. It may not look much like Figure 15 (the 'growl' section from last month's Synth Secrets) but, with skilful application of the mod wheel, the result can be much the same.

## Amp & Filter Together

Ignoring the effect of the filter modulation, let's now consider the combined action of the Loudness Contour and Filter sections, and see what happens to any given harmonic within the spectrum of the initial sawtooth wave.

You know that, at the instant that you press a key, the low-pass filter is almost closed (but not completely, because keyboard tracking is set to 100 percent On). Therefore, the fundamental plus a handful of the lowest harmonics pass immediately through to the audio signal VCA, and their rise time is determined by the Loudness Contour's Attack speed of 100mS. Then, because the filter opens more slowly than the amplifier (the filter's Attack is set to 600mS) the higher harmonics are let through one by one over the course of about half a second. Furthermore, different harmonics are emphasised as the cutoff frequency is swept, all of which is as we would expect in a real brass instrument. Figure 16 on the next page (which shows the response of real brass instruments) shows a simplified representation of this, and confirms that the Loudness Contour and Filter are set correctly.

## Pitch Modulation & Noise

At this point, it would be useful to add the shaped noise described last month.

▶

Unfortunately, while you can add noise in the Minimoog's Mixer, it lacks the formant shaping of the turbulent noise in a real brass instrument, and sounds very unnatural. Consequently, it is best omitted.

It would also be beneficial to add delayed pitch modulation (vibrato), but I have run out of facilities... the Minimoog has only one modulation source, and no spare contour generators or VCAs, so it simply isn't capable of this. Sure, I could sacrifice the growl for a steady vibrato, but I can't have growl and vibrato simultaneously. Or can I...?

Let's return to the performance controls in Figure 6 where, next to the Mod wheel, you'll find the Pitch wheel. With a bit of practice, it's possible to introduce vibrato manually, by moving this wheel backwards and forwards very slightly! This isn't as strange as it may sound — it's not dissimilar to what a guitarist does by bending strings, after all — and it can produce vibrato that is much more natural than that generated using an LFO. Indeed, with practice, you can alter the amount and speed of the vibrato, as the music requires. You can also move the wheel more dramatically to imitate the slides of a trombone, but that would be quite inappropriate for a trumpet and the other brass instruments.

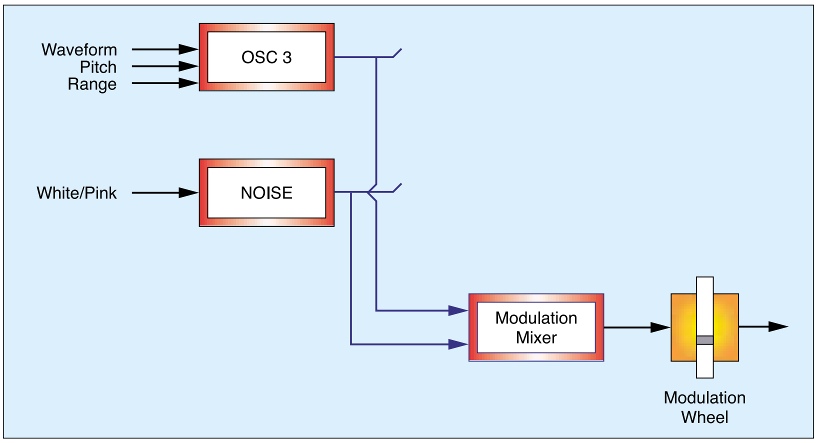


Figure 14: Using the Mod wheel to control the modulation level.

## The Resulting Patch

Figure 17 (below) shows all the elements I've described, and there's no reason why you shouldn't walk up to a Minimoog and patch it as shown. Your results won't be *exactly* the same as mine, because in no two Minimoogs are the voice circuits identical, nor are their knobs calibrated identically. So, be prepared to tweak things a little.

Looking back at the Minimoog block diagram in Figure 1, it's interesting to eliminate all the parts that are unused, and see how the switch positions shown in Figure 17 have configured the instrument (see Figure 18 on the next page). As you can see, many elements of the synth are unused, including Osc2, the pitch modulation, the slew generator, the Release generators, and the external input. This has simplified matters considerably, so it shouldn't be too hard to relate the switch settings in Figure 17 to the block diagram in Figure 18.

Perhaps even more intriguing is Figure 19, right underneath Figure 18, which shows just how little of Figure 2's idealised brass patch is recreated on a Minimoog. I've left the blank spaces from which I've removed all the unused modules, just to emphasise the limitations imposed by the Minimoog.

Given this, it's astonishing how good a well-patched Minimoog can sound. Sure, its limited voicing and even more limited performance capabilities will ensure that it never sounds like a real brass instrument, but with sympathetic EQ and a suitable reverb, it's remarkable how close you can get. This is especially true when playing lower-pitched brass sounds such as trombones and tubas, because the ear is less sensitive to the nuances of their sounds.

## Other Patches

I thought that it might be interesting to compare my brass patch to those published in 1977 in *Minimoog Sound Charts* by Tom Rhea. This book, which also contains patches by Keith Emerson and Rick Wakeman, was produced by Moog's '70s parent company Norlin Music, and was included with later Minimoogs; many people consider it to be the definitive guide to the synth. But if you inspect Figure 20, you'll see that Rhea's trumpet patch is very different from the one I've created.

Most obviously, Rhea has used all three oscillators as sound sources, tuned in unison so that (in his words) you can "add ▶

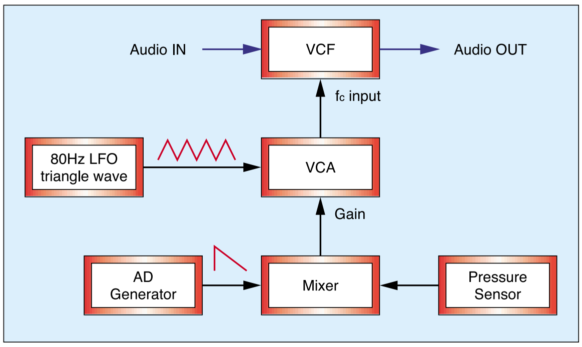


Figure 15: Adding 'growl' to the sound.

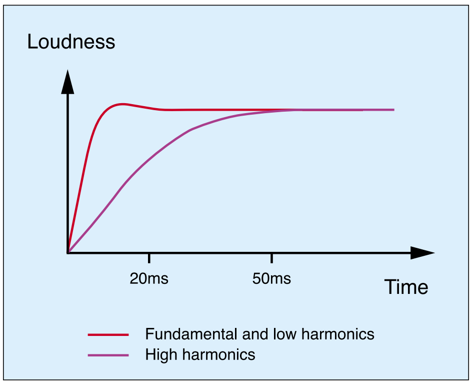


Figure 16: Higher harmonics take longer to 'speak' than lower ones.

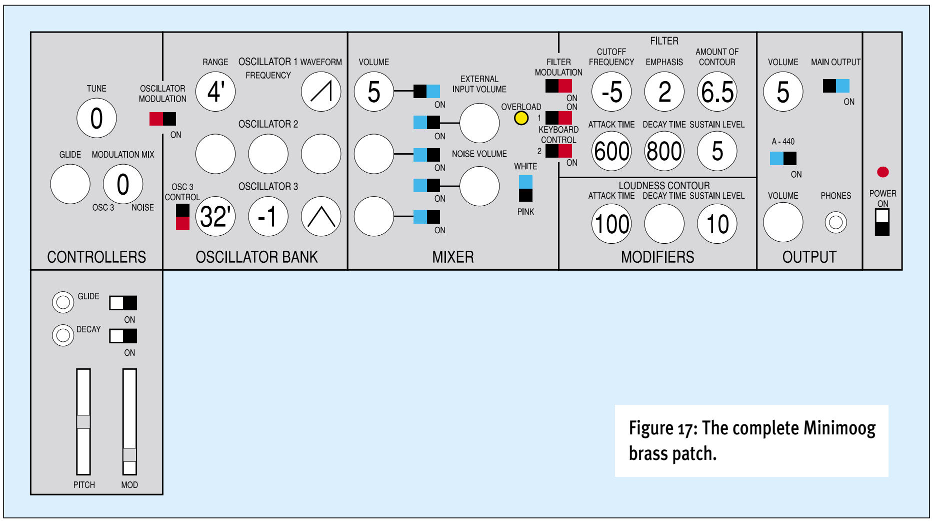
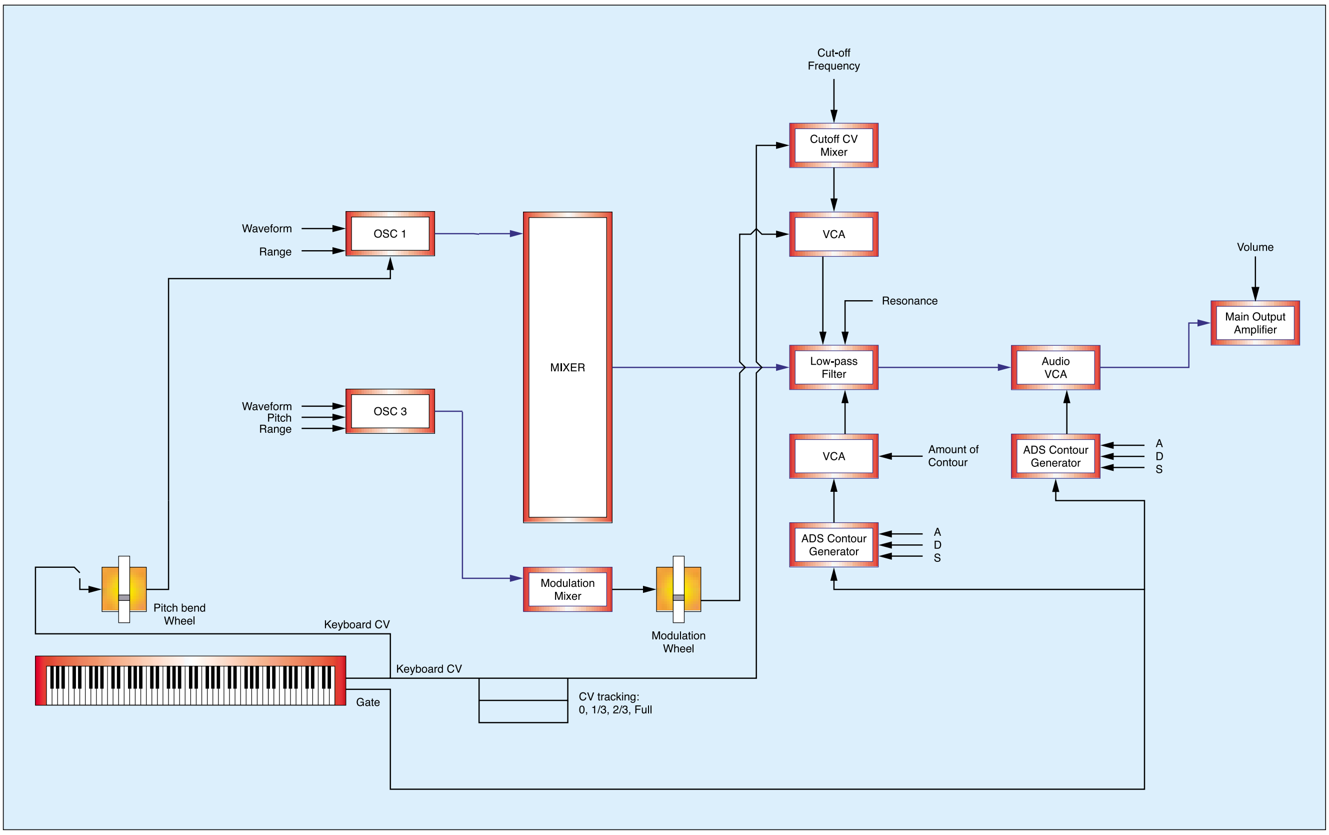


Figure 17: The complete Minimoog brass patch.

oscillators for progressively 'Fatter' tutti sounds". This means that there is no modulation (other than any manual vibrato you add using the pitch wheel).

If you study the Loudness Contour and performance panel, you'll see that Rhea's Attack setting is a little slower than mine, and that he has added a short Decay once the note is released. But these are not huge changes…  much more significant are the changes in the Filter settings. There's no filter modulation, keyboard tracking is

Figure 18: The routings used for the Minimoog brass patch.



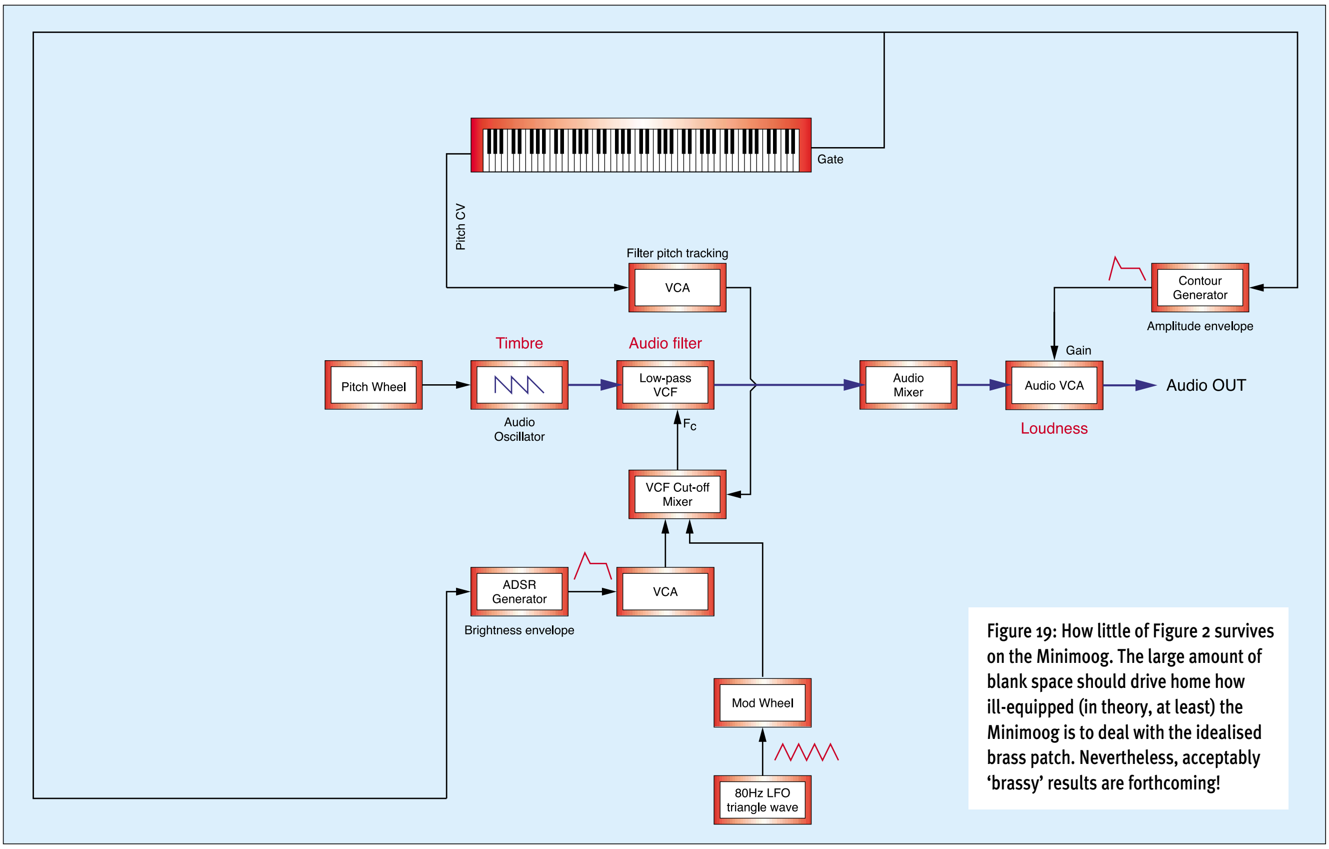


Figure 19: How little of Figure 2 survives on the Minimoog. The large amount of blank space should drive home how ill-equipped (in theory, at least) the Minimoog is to deal with the idealised brass patch. Nevertheless, acceptably 'brassy' results are forthcoming!

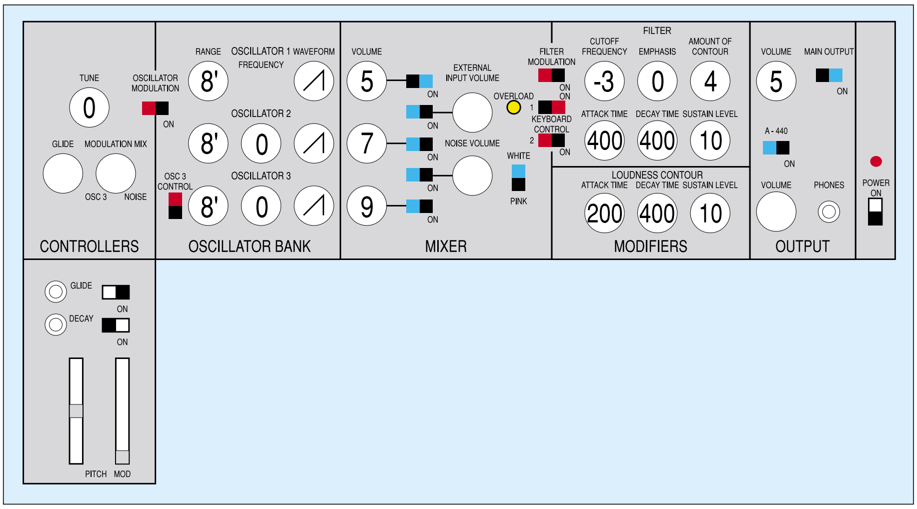


Figure 20: Tom Rhea's Trumpet from the Minimoog patch book.

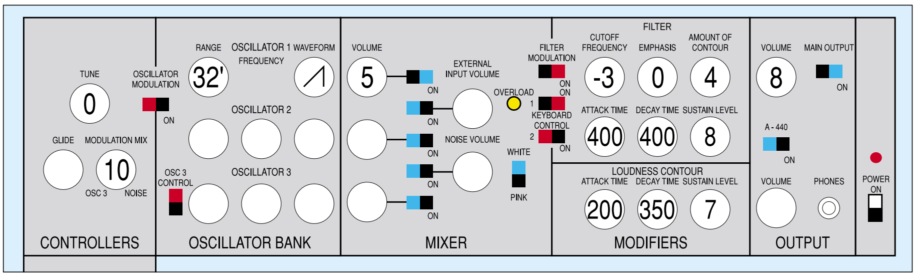


Figure 21: Tom Rhea's Tuba patch. The Performance panel has been omitted to save space (the settings on it are the same as in Figure 20).

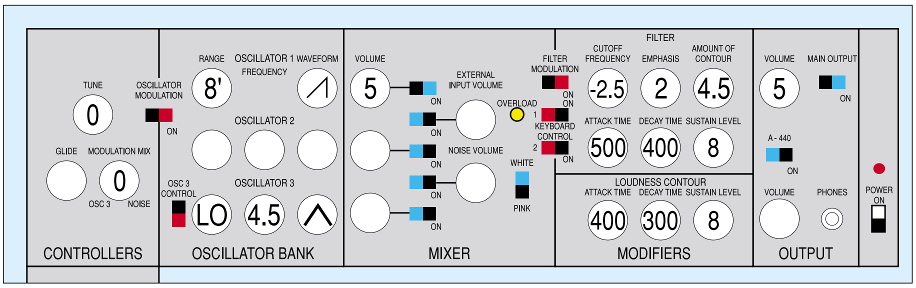


Figure 22: Tom Rhea's Jazz Trombone. Once again, the settings on the omitted Performance panel are the same as in Figure 20.

just 66 percent, the Attack is faster, the filter cutoff starts at a higher frequency, there's no emphasis, and the Amount Of Contour is lower. To some extent, two of these cancel out — the more open filter and the lesser envelope — but these settings do not really conform to the theory laid down in parts 24 and 25 of this series.

Consequently, the book's patch does not sound as realistic as mine. This may sound a little arrogant, but I have based my patch on firm scientific principles, so it would be surprising if it did not retain the essential character of the brass instrument. I find Rhea's sound to be somewhat muted, and feel that it lacks the movement introduced by the filter modulation that I have used.

In contrast, Rhea's Tuba patch (see Figure 21, above) adheres much more closely to the theoretical principles discussed here over the past couple of months. Rhea's Tuba retains the less aggressive filter contour used on his trumpet, as well as the short Decay at the end of the note, but employs a single sawtooth oscillator and filter modulation. The most interesting part, however, is the source of the modulation. Instead of using Osc3 (and therefore risking FM side-bands) he uses the noise generator to roughen up the filter. This proves to be extremely effective. He also recommends that players experiment with the Amount Of Contour and Cutoff Frequency knobs to create brassier or more muted effects.

Finally, Rhea's Jazz Trombone patch in Figure 21 (below left) contains many of the elements described earlier. The envelopes are similar, the single sawtooth oscillator is the same, and he uses Filter Modulation. Here he uses Osc3 as a true LFO (whereas I have used it as an audio frequency modulator), and he adds Oscillator Modulation (vibrato). Personally, I find it's very difficult to limit the vibrato to a reasonable level using the Mod wheel, but this is nonetheless an effective patch when played with care.

## Handle With Care

There are a couple of points that need restating before I finish. Firstly, bear in mind that your appreciation of a trumpet's sound may be very different to mine. Indeed, you might be thinking Royal Philharmonic, while I'm thinking Satchmo. This means that all of the settings shown in this article are guidelines. Nevertheless, however you manipulate the sound, it must retain the common elements shown in the diagrams here. If you stray too far from these settings — perhaps by changing the oscillator setting so that it produces a different waveform, or by altering the envelopes so that they exhibit instantaneous attacks, or by adding a high degree of filter emphasis — none of the resulting patches will sound remotely brassy.

The second concerns performance. You may have access to the perfect brass patch, or to a perfectly recorded trumpet sample, or even to the physically modelled brass patches on a Yamaha VL7 or a Korg Z1. However, none of them will sound authentic unless you play them in a manner that is sympathetic to the original instrument. Some synthesists — Wendy Carlos is an excellent example — manage to coax remarkably lifelike performances from their synths. You and I, on the other hand, may find ourselves unable to approach the same level of authenticity, even when playing the same patches on the same instruments. When this happens, our natural inclination is to blame the equipment, the outboard effects, or the person who gave you the patch settings. But before you do this, consider your playing technique. Synthesis is not just about sounds; it's also about performance. And no amount of signal-routing diagrams can help you with that. SOS

# synth secrets

## PART 27: ROLAND SH101/ARP AXXE BRASS SYNTHESIS

**Gordon Reid** concludes his attempts to adapt an idealised analogue brass patch so that it can be programmed on real synths. This month, he looks at the Roland SH101 and ARP Axxe.

Last month, I used the Moog Minimoog to create a patch designed to represent — as far as possible — the acoustic principles of brass instruments, as discussed in Synth Secrets 24 and 25. The result was a range of sounds that, despite several compromises, exuded the *essence* of brassiness, if not the exact timbre.

Unfortunately, not many people are fortunate enough to own a Minimoog. Only 12,000 or so were ever made and, on those rare occasions that one appears for sale, the price tag is often in the region of £1,000... which puts it beyond the reach of most players. Cheaper, less endowed synths are far more common, so you're much more likely to own, for example, a small Roland than any Moog. But does this mean that brass sounds are the preserve of the fortunate few? Not a bit of it! This month, I'm going to take what is perhaps the most popular analogue monosynth of our time — the Roland SH101 — and apply the same principles as last month.

### Comparing The Roland SH101 & Minimoog

Figure 1 (below) shows the top panel of an SH101 with all its controls set to zero. It doesn't look much like the Minimoog from last month, does it? That's not surprising... much of its architecture is unlike that of the Moog. On the other hand, there's a surprising amount that's similar, and some that is identical. So, before going any further, let's compare the two synths, and get to grips with the problems you might encounter as you try to translate a patch from one to the other.

Starting on the left, and ignoring trivia such as tuning knobs and On/Off switches, both synths offer a Modulation section. However, whereas the Minimoog limits you to using Osc3 and/or the noise generator as a modulator, the SH101 offers a dedicated LFO. This offers just four modulation waveforms (compared to the six on the Minimoog's oscillator) and is strictly low frequency... you cannot use it for two-operator FM synthesis as you can Osc3 on the Minimoog. Furthermore, noise is an LFO waveform, so you can't mix this with the cyclic waveforms as you can on the Moog. But on a positive note, the SH101 has a modulation option called 'Random', which is a Sample & Hold generator clocked at the LFO rate. This makes possible a number of effects that you cannot obtain from the Minimoog.

Moving to the right, you come to the synths' Oscillator sections. The major difference here is obvious: whereas the Minimoog has three audio frequency oscillators, the SH101 has just one. However, Roland has minimised the shortcomings of this by allowing you to modulate the pulse width of the square waveform using the LFO or the envelope generator. This means that the SH101 can produce a range of rich, chorused sounds that you can't obtain from the Minimoog. Furthermore, whereas the Minimoog has just a toggle to control the amount of Oscillator Modulation (vibrato), the SH101 has a dedicated control that allows you to

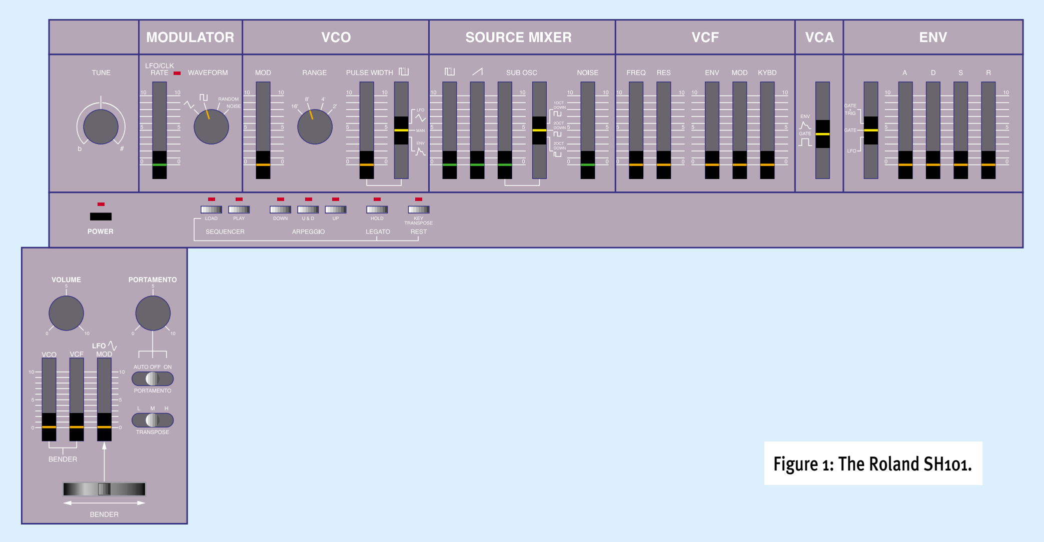


Figure 1: The Roland SH101.

apply as much or as little as you require.

Because the SH101 has just a single oscillator, you might think that it needs no Mixer. However, the oscillator in the Roland produces three waveforms simultaneously — a sawtooth, a pulse wave, and a square sub-oscillator — and the Mixer is where you recombine them. With a separate noise generator as well, the SH101's architecture may not be as flexible as the Minimoog's, but it's not shoddy either.

Next come the synths' 24dB-per-octave low-pass filters. Like the Minimoog, the SH101 offers control over the cutoff frequency, the resonance ('Emphasis'), and the amount of contour, but it also has variable controls for modulation and keyboard tracking. Unfortunately, whereas the Moog's filter has a dedicated ADSD contour generator, the Roland's has to share its single ADSR with the audio VCA.

This brings us neatly to the VCA and its associated contour generator. On the Moog, the amplifier's contour generator is another ADSD gated by the keyboard. In contrast, the Roland has a true four-stage ADSR that can be triggered by the keyboard (Gate or Gate+Trigger) or by the LFO. Furthermore, you can disconnect the ADSR from the VCA by placing the switch into the 'Gate' position, thus leaving the contour generator free for purely VCF duties.

The last set of controls lies in the performance panel found to the left of the keyboard on both synths. The Minimoog has Pitch and Mod wheels, but the Roland is somewhat more flexible. You can set the maximum amount of pitch-bend and filter modulation produced by moving its bender controller in a left/right direction, and set the maximum amount of LFO modulation produced when you push it away from you. You have more control over portamento, too. Whereas the Moog offers a Glide rate control and an On/Off switch, the Roland adds an 'Auto' function, which applies portamento only when you hold two keys simultaneously.

The SH101 has one more trick up its sleeve. When you set the VCA mode to 'Gate', the synth responds to its keyboard in the same way as the Minimoog does. It is low-note priority and when you play, it only retriggers its contour generator after you have released all previous keys. But when you set it to 'Gate+Trig' it becomes last-note priority, and generates a trigger every time you press a note, or when you release a note to allow an older one to sound again. This type of response is of huge benefit for certain types of playing. Unfortunately, it is not available on the Minimoog.

## The Brass Patch On The SH101

Now that you appreciate many of the differences between the two synths, you're in a position to think about defining the idealised brass sound

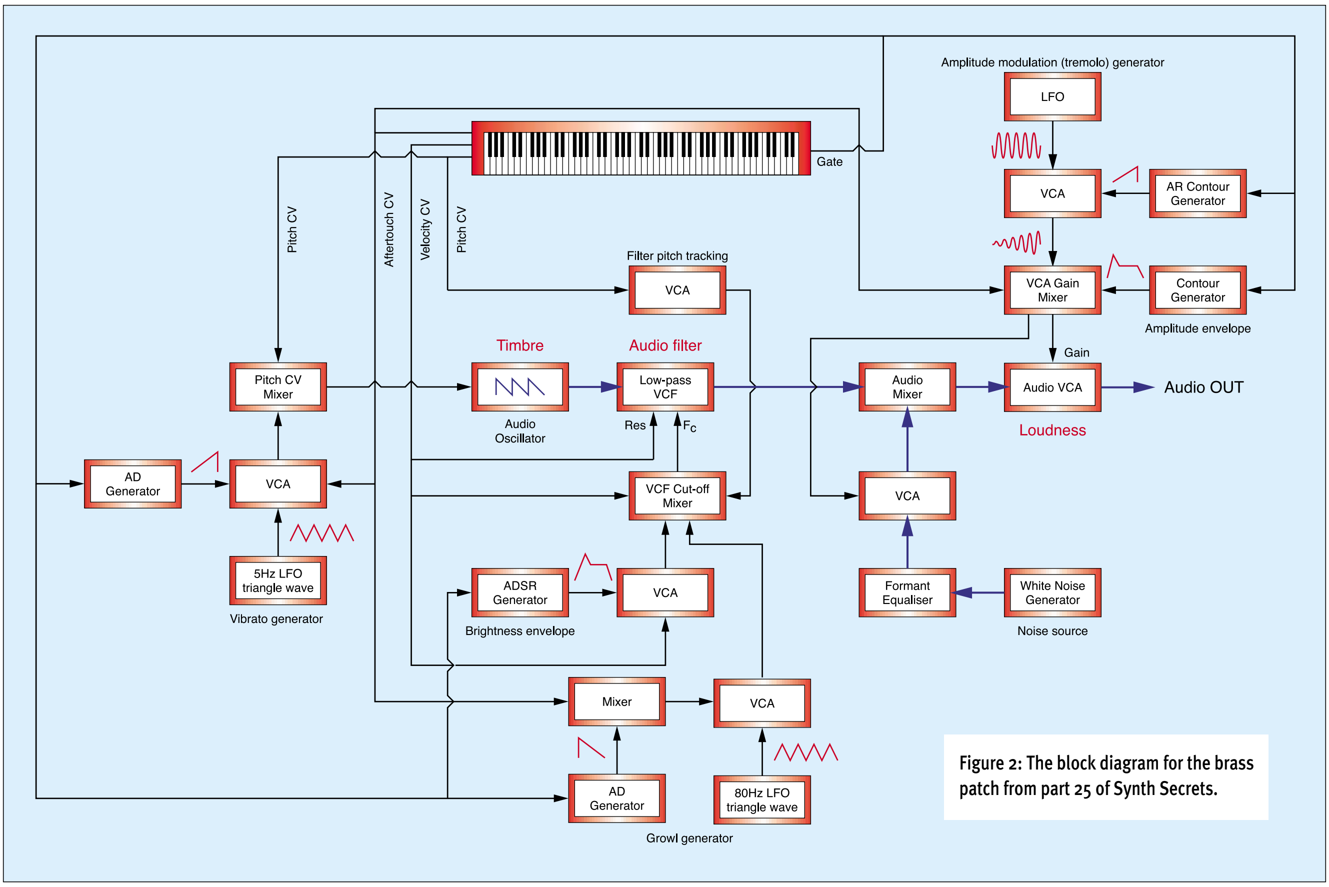


Figure 2: The block diagram for the brass patch from part 25 of Synth Secrets.

▶ using the more limited (single-oscillator, single-contour-generator) architecture offered by the SH101. Let's start by referring back to the block diagram for a brass patch, first shown in part 25 of Synth Secrets (see Figure 2, below).

Clearly, the SH101 has far too few component modules to recreate this in full, so I'll have to restrict myself to the most important elements of the sound. This means that the delayed vibrato to the left of Figure 2 will have to go. Similarly, the shaped noise is a goner. As for the tremolo and some of the complexities of the growl generator... sorry chaps, but there's no place for you, either. So what *can* I do?

Fortunately, the idealised brass patch requires only one oscillator, so the SH101's limitation in this area doesn't affect the sound. I can select the 4' range in the VCO section, and raise the sawtooth volume fader in the Mixer to allow this waveform to enter the signal path. At the same time, I must make sure that the pulse, sub-oscillator and noise faders are at zero, or these waves will change the fundamental nature of the sound, making it unsuitable for brassy timbres (see Figure 3, above right). I'll also set the Mod control in the VCO section to zero; like last month's sound, this patch will have no LFO-driven vibrato.

Since I'm using only the sawtooth wave, I can ignore the Pulse Width controls in the VCO section (they affect only the pulse wave) and the sub-oscillator waveform selector in the Mixer.

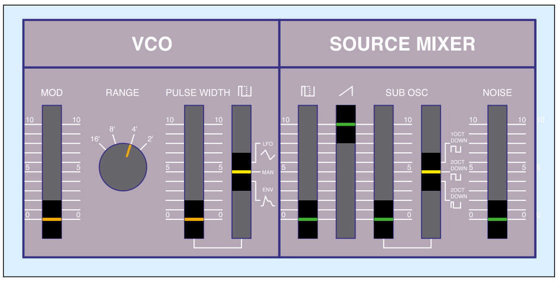


Figure 3: The SH101 oscillator and mixer settings.

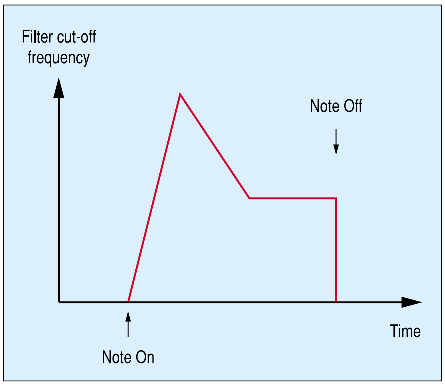


Figure 4: The desired filter contour for the SH101 brass patch.

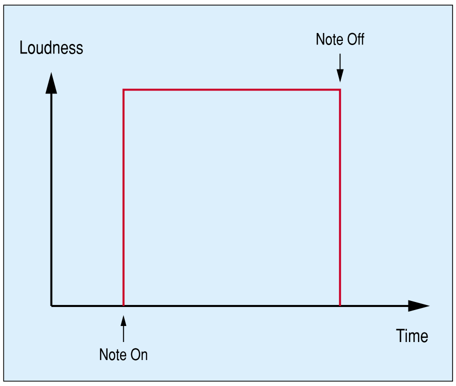


Figure 5: The loudness envelope generated by connecting the Gate to the VCA.

## Shaping The Waveform — Loudness & Tone

Again, it's time to filter and shape the sawtooth waveform. You'll remember that last month I used the contour generator in the Loudness Contour section to generate an envelope with a short Attack and instantaneous Release. At the same time, the contour generator in the Filter section generated a more 'shaped' contour to determine how the tone changed over time (see Figure 4, above right) Well... this can't be done on an SH101, because it only has one contour generator.

However... I've already mentioned that you can disconnect the SH101's VCA from its contour generator, and connect it directly to the Gate from the keyboard. This means that I can create the loudness contour shown in Figure 5, which is close enough to the Minimoog contour to give the result I want. I have shown the control panel setting for this in Figure 6 (see right).

Returning now to the filter contour, the SH101 has a distinct advantage over the Minimoog... it has a dedicated Release stage in its contour generator. You might think that there's no point in setting the release slider to anything other than zero for this patch, since the gain of the VCA will return to zero the moment you release the key, However, the SH101 envelope is very rapid, and you can hear it snap shut when the Release is set to zero (many novices complain that the contour generators on their synths generate 'clicks' when the Attack and/or Release controls are set to low values, not realising that this is a compliment to the electronics, not a fault). Setting the Release fader to '2' gives a nice, smooth tail to each note.

This brings me to some thoughts about how brass players move between notes. If they are tonguing notes, there will be a short break between each, and you must play the keyboard in a slightly staccato style to emulate this. However, if the player is using a valve instrument, and presses a valve to change pitch, the transition will be smoother. (Actually, it could be very uneven because the player's lip tension is no longer appropriate to the length of tube... but that's not a discussion for today.) Anyway, the important point is that there will be no re-triggering of the notes' contours in this case. This means that I must set the filter contour to respond to Gate only, and not Trig. The Env settings are therefore as shown in Figure 7 below.

Of course, all these settings will be useless if the VCF itself is not set up correctly. It needs to be almost closed at the start of the note, and should ▶

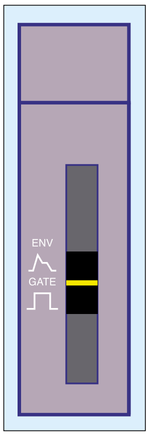


Figure 6: Using the Gate to determine the VCA envelope.

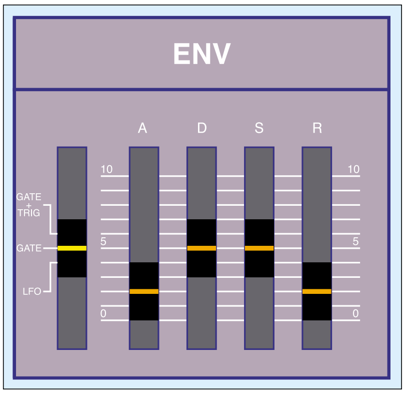


Figure 7: The Env settings for the filter contour.

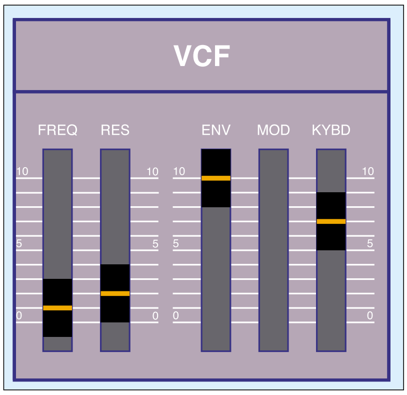


Figure 8: The VCF settings.

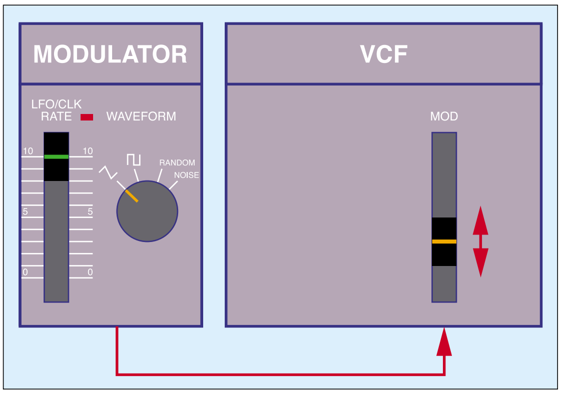


Figure 9: Applying rapid modulation to obtain growl at the start of the note.

open a great deal during the Attack phase of the envelope. This means that the Freq slider in the VCF section must be close to zero, and the Env fader must be at or near its maximum. Furthermore, as you should know from the theory explained two months ago, there must be just a touch of resonance to create the correct harmonic profile, and the cutoff frequency must track the keyboard at a little less than 100 percent. If I ignore the Mod fader, this allows me to define the VCF section as shown back in Figure 8.

Now for some bad news. Like the Minimoog, the SH101 has only one source of modulation. This means that I must make a choice. Do I want to use the Modulator to add gentle vibrato to the patch, or the more dramatic 'growl' that is so effective in brass patches? Clearly, by the way I've worded that sentence, it's the latter. But, since the SH101 lacks the EGs and VCAs needed to control the growl, I will have to do this manually, using the Mod fader in the VCF, adding modulation at the start of the note and removing it as the note approaches its steady state. I find that an initial setting of about 60 percent works well, reducing this to 0 percent by the time the filter contour reaches the Sustain Level (see Figure 9, left).

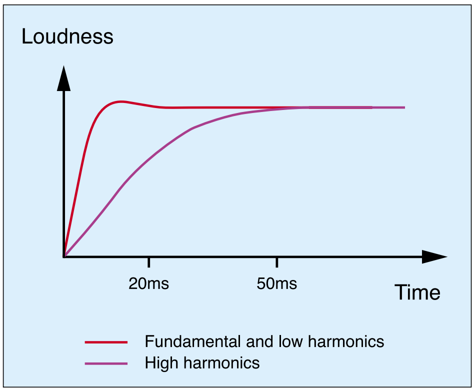


Figure 10(a): The ideal rise times for lower and upper harmonics.

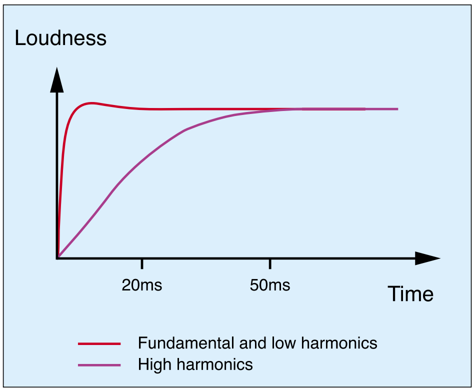


Figure 10(b): The SH101 response for lower and upper harmonics.

"Whether you're programming a '70s Minimoog, an '80s Roland SH101, a '90s Nord Lead or a 21st-century Access Virus Indigo, the principles for a given sound remain identical."

Of course, I also need to set the Modulator appropriately. Because the SH101 Modulator is purely an LFO, I do this by setting it at its highest rate. As for the waveform... I use the triangle wave, as I did on the Minimoog.

Last month, I showed that the Attack of the Loudness contour, coupled with the slower Attack of the Filter contour, allowed the harmonics to enter the sound at close to the appropriate rates, as defined by the analysis of real brass instruments (see Figure 10(a) below). This month, I don't have the luxury of the VCA contour so, as shown in Figure 10(b), the lower harmonics enter rather too quickly (this is the consequence of the square loudness contour shown in Figure 5.)

The audible consequence of this is that the start of the note is rather less authentic than it was on the Minimoog. But... (and it's a big 'but') two things save the day. Firstly, no VCA responds instantaneously, so there is still a slight lag in the rise time of the loudness contour. Secondly, any residual deficiency can be masked using the growl effect described above.

Moving on, it would be nice to be able to add the shaped noise described in part 25 of Synth Secrets. Unfortunately (and in common with the Minimoog's noise generator), the SH101's noise generator lacks the formant shaping of the turbulent noise in a real instrument, and sounds very unnatural. As on the Minimoog, it is best omitted.

It would also be beneficial to add delayed pitch modulation (vibrato) to the patch. And, yet again, I have run out of facilities... the SH101 has only one modulation source, so it isn't capable of this. At a pinch, I could try the same trick as last month, and use the pitch bender to add vibrato manually. If I set the VCO Bender fader to a small value, I can then move the Bender itself from side to side to create the desired effect (see Figure 11, right).

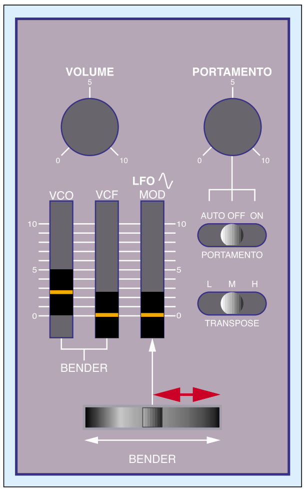


Figure 11: Adding a little manual vibrato.

Unfortunately, this will be at the expense of the growl, because you can't play the note, manipulate the bender, and manipulate the Mod fader simultaneously. So the best you can do is play the notes and add *either* growl or vibrato, depending upon the requirements of the music.

Putting everything together into a single patch, Figure 12, on the next page, combines everything I've described. It looks very different from the Minimoog patch shown towards the end of last month's instalment of Synth Secrets, but the

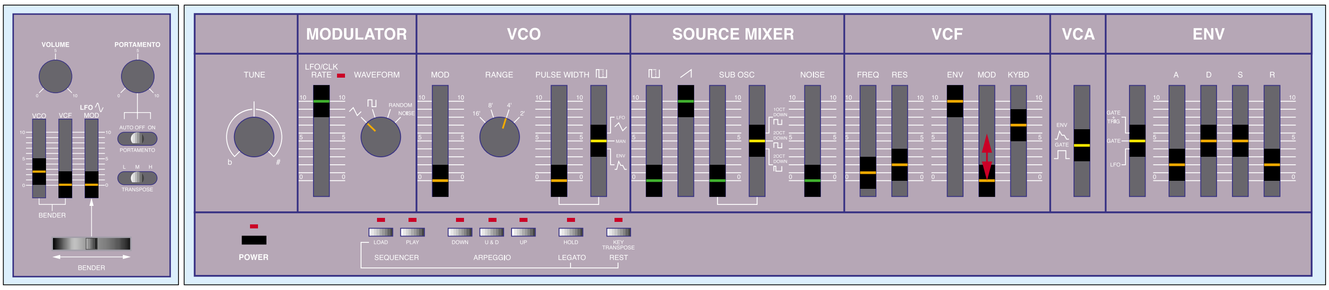


Figure 12: The SH101 brass patch. Note: the performance panel control has been placed at the right end of the diagrams on this page for clarity.

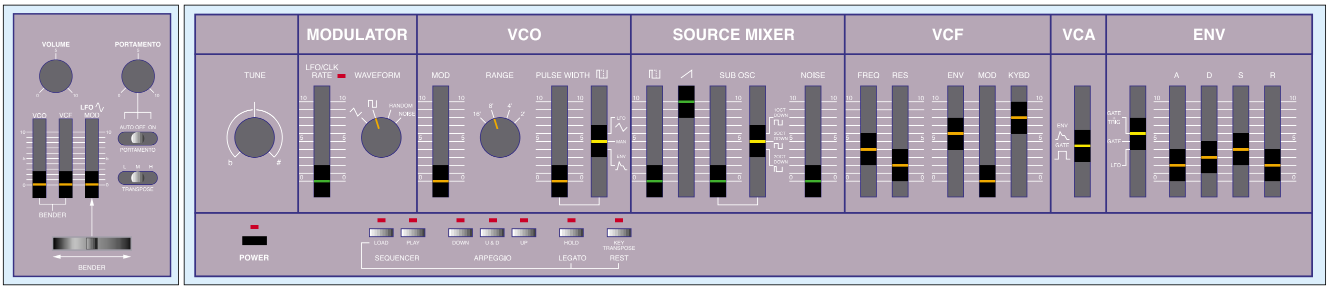


Figure 13: Roland's Trumpet patch for the SH101.

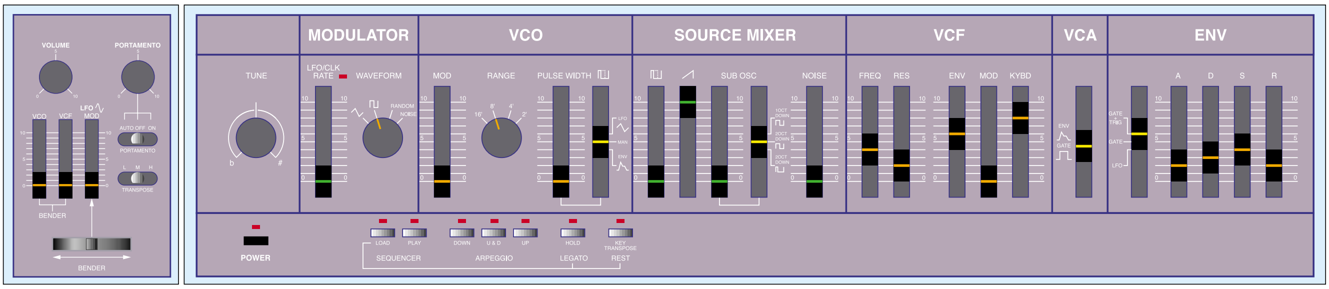


Figure 14: Roland's Tuba patch for the SH101.

▶ audible result is nevertheless as similar as it is possible for two such different synths to be. Moreover, although the SH101 is unable to recreate much of the patch in Figure 2, set up carefully and played sympathetically it can still sound remarkably brassy.

## Other SH101 Patches

Before leaving the SH101 behind, I thought that it would again be instructive to analyse a couple of the factory patches; one good, one bad. Let's start with Figure 13 above; Roland's Trumpet patch in the original SH101 manual.

This is a very disappointing sound. The Attack/Decay stages of the Env are too short, the amount of Env control in the filter is too low, and the higher initial cutoff frequency allows too many harmonics through when you first press a key. Furthermore, there's no modulation, so there's no movement in any portion of the note. Yurgh!

The Tuba patch from the SH101 manual fares much better, and introduces another concept: that of adding different waveforms to achieve a particular timbre. Take a peek at the Mixer section in Figure 14 above. You'll see that the sawtooth is present at 60 percent of its maximum, but that there's also a square wave sub-oscillator present, one octave down and at 100 percent of its full loudness. The result is the harmonic spectrum shown in Figure 15 (right), and the waveform shown in Figure 16 on the next page.

As you can see, while remaining sawtooth-like, both the spectrum and the waveform are more complex than those of a simple sawtooth and, of course, the timbre changes appropriately. If you have access to an SH101, listen to the patch with the sawtooth alone (it lacks body) and then to the square wave sub-oscillator alone (it sounds hollow, and not at all brassy). In this patch, the combination of the waveforms defines the sound, almost as much as the filter and amplitude ▶

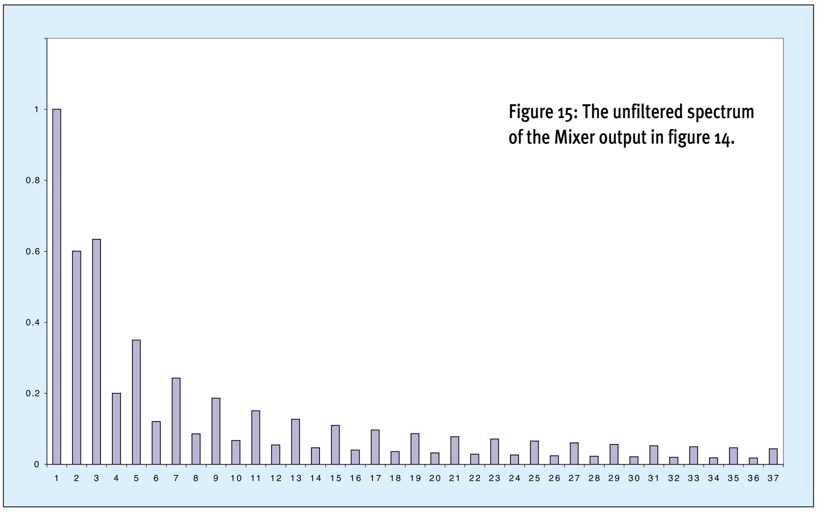


Figure 15: The unfiltered spectrum of the Mixer output in figure 14.

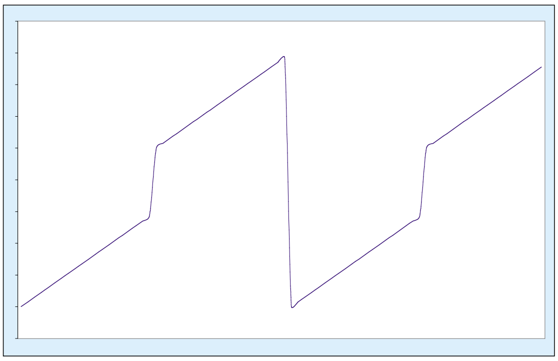


Figure 16: The waveform described by the spectrum in Figure 15.

settings. This is something that will come up again in later parts of this series.

## The Brass Patch On The ARP Axxe

Figure 17 shows the control panel of the ARP Axxe, another low-cost, single oscillator monosynth. At first sight, this appears to be very different from the SH101, but look more closely. There's just one voltage-controlled oscillator, which produces both sawtooth and pulse waveforms simultaneously. There's also a noise generator, variable pulse width on the pulse (square) waveform, and pulse-width modulation courtesy of the LFO and the ADSR. The LFO produces sine, square and S&H waveforms. You can modulate the 24dB-per-octave, resonant low-pass filter using the LFO and/or the contour generator, and it will track the keyboard in any amount from 0 percent to 100 percent. There's just a single contour generator, and it's a four-stage ADSR... The list goes on and on. Sure, there are differences between the Axxe and SH101 too. For example, the Axxe has no sub-oscillator, and like the Minimoog, it possesses an external signal input, not included on the SH101. Nevertheless, the important elements of the SH101 and the Axxe are the same.

So why do they look so different? The reason is simply one of presentation. The controls and panel graphics for the Axxe are based on ARP (ie. American) designs that first appeared on the ARP 2600 in 1970. In contrast, the SH101's panel is a development of the Japanese Roland Juno 6, which first hit the streets in 1981. Same facilities, different countries, different eras; hence the different appearance.

Now, ignoring these superficial differences, turn your mind back to brass patches. Logic tells you that, if the two synths' facilities are the same, I should be able to set up a brass patch on the Axxe, simply by remembering how I did so on the SH101.

Locating the Audio Mixer section on the Axxe, I find the sawtooth waveform produced by the VCO, and raise this to its maximum. At the same time (and for the same reasons as before) I must ensure that the square waveform and noise faders are at zero.

Next, I locate the five faders that control the Voltage Controlled Filter, and raise the VCF Freq (initial cutoff frequency) and the Resonance faders slightly. Then, I set the ADSR tracking fader to a high value, and the Kybd CV tracking to a moderate value. The sine wave LFO modulation fader can be left at zero.

Moving to the right, I raise the initial VCA Gain in the Voltage Controlled Amplifier. However, I leave the ADSR fader at zero. This disconnects the VCA from the ADSR, just as on the SH101.

Finally, at the far right of the panel, I set the ADSR Envelope Generator to something approximating the SH101's 20 percent, 50 percent, 50 percent, and 20 percent values.

Figure 18 opposite shows the patch thus defined. If I now stop and play the Axxe, I'll find that I have something that sounds *similar* to the SH101 and Minimoog brass patches but, in a number of ways, isn't quite right. And, for some reason, there's some sound leaking through all the time... Arghh! The note never dies!

## Final Tweaks

Let's deal with the big problem first. There are apocryphal stories of ARP 2600 owners who, in the early days, contacted ARP to say, "Wow! It's amazing... but how do you make it *stop*?" The reason lies in the combination of the VCA Gain and VCF Freq sliders. If the first of these is greater than zero, the VCA is always amplifying (ie. passing) any audio signal received at its input. This means that the only way to silence the sound is to remove *all* its harmonics using the filter. Therefore, I must reduce the cutoff frequency to zero between notes if silence is to reign, and I can do this by moving the VCF Freq slider to zero.

Unfortunately, this contravenes one of the principles of the idealised brass sound... that the fundamental should pass as soon as you play a note. So maybe a better compromise would be to reduce the VCA Gain to zero, and use the ADSR to open and close the amplifier. Now, the amplifier is controlled by the ADSR and, again, silence will reign between notes (see Figure 19, right).

Playing the Axxe patch again, it still isn't quite right. To understand this, you must remember that all synths are not created equal. The circuits within different models will respond to the controls in slightly different ways. So I must tweak the settings to obtain the right results on the ARP. In this case, I find it pleasing to increase the Decay time, reduce the Sustain Level, and reduce the ADSR CV in the VCF... all of which emphasise the initial parp of the brass sound.

One thing I can't do, however, is produce the filter rasp that was so successful on both the Minimoog and the SH101. This is because the LFO has a maximum frequency of just 20Hz, which is not fast enough to create the desired effect. However, the last thing I want is a static, boring sound, so I'll use the LFO to introduce a gentle vibrato — something that also sounds good on brass instruments.

"...if the two synths' facilities are the same, I should be able to set up a brass patch on the Axxe, simply by remembering how I did so on the SH101."

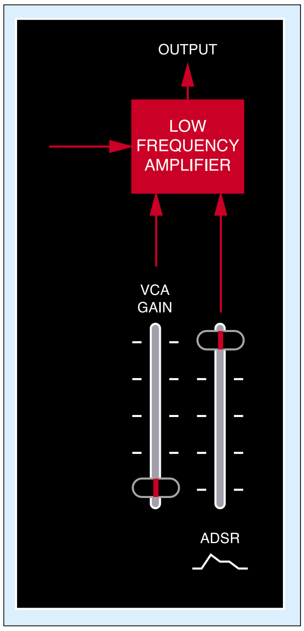


Figure 19: Silencing the ARP Axxe.

I *could* do this by setting the LFO rate to around 5Hz, and by raising the LFO sine wave CV fader in the VCO (it's the second fader from the left). However, there's a better solution. This 'Mark 2' Axxe has Proportional Pitch Controls (PPCs) shown on the far left of the panel. When you press down on the middle one of these pressure-sensitive pads, it adds modulation, just like an aftertouch-sensitive keyboard. The PPC therefore allows you to add vibrato in a realistic fashion... it's far better than the fixed amount of low-frequency modulation that would have been introduced using the LFO slider.

The final Axxe brass patch appears in Figure 20 below, and you won't be surprised to learn that it sounds very similar to the both the Minimoog and SH101 patches. OK, so each of these synths has an individual character, and there are many analogue *aficionados* who could distinguish between them. But that's not the point. The important thing here is that I've succeeded in programming the same sound on all three instruments. Indeed, you may not realise it, but you've learned an important lesson over the past couple of months. It's this:

*Once you've learned how to create a brass patch on one synth, you can recreate it on any synth capable of doing so.*

So, whether you're programming a 1970s Minimoog, a 1980s Roland SH101, a 1990s Nord Lead or a 21st-century Access Virus Indigo, the principles for a given sound remain identical. Once you understand what it is that defines 'brassiness', you can program the equivalent patch on any subtractive synth. Neat, huh? SOS

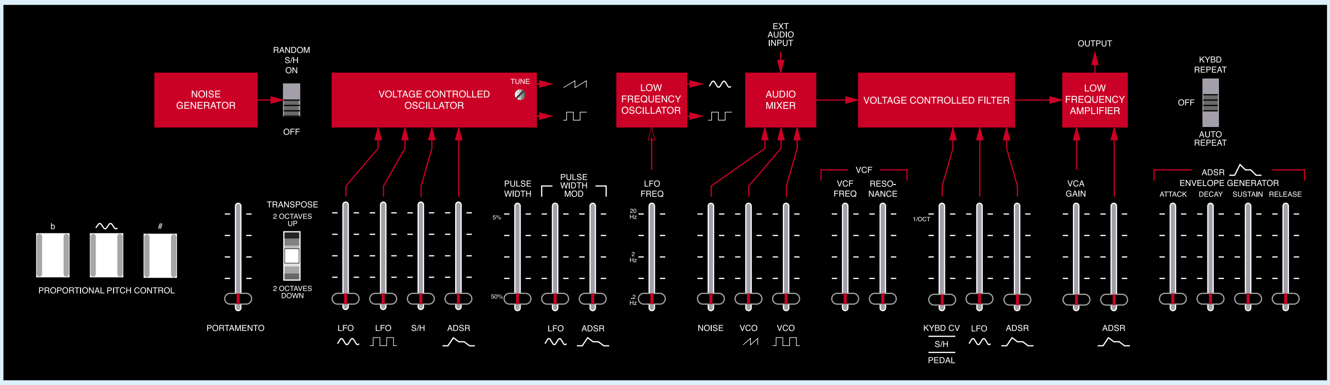


Figure 17: The ARP Axxe.

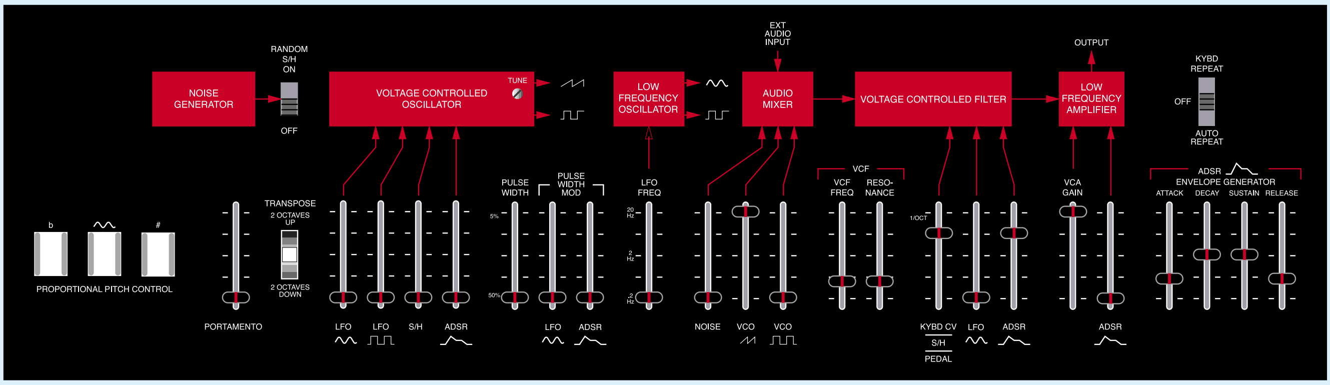


Figure 18: The idealised trumpet sound recreated on the ARP Axxe.

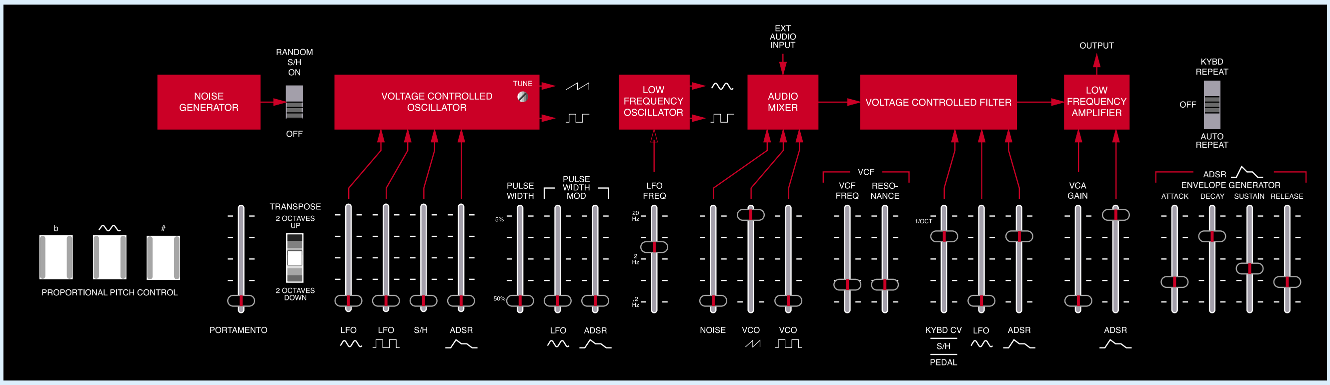


Figure 20: The final ARP Axxe brass patch.

# synth secrets

## PART 28: SYNTHESIZING PLUCKED STRINGS

Having dealt exhaustively with the mechanics of brass instruments and how to go about synthesizing them, **Gordon Reid** turns to instruments that use plucked strings to generate their sound, taking the complexities of the acoustic guitar as an example...

If you've been following Synth Secrets over the past four months, you'll have studied the physics of brass instruments, and seen how analogue synthesizers can recreate the essence of brass sounds. You're no doubt wondering whether other instruments can be analysed in the same way. The answer is yes, although if you thought brass instruments were complex sound-producing entities, you may be in for a shock this month, as I turn my attention to the principles of plucked strings and resonant bodies, and consider how these principles might help us to synthesize the sound of the acoustic guitar.

### Part 1: The String Itself

I'll start, as I did in part one of Synth Secrets, by considering the vibration of a stretched string. By now, you're all familiar with the fact that such a string is capable of vibrating at all the frequencies that comprise the harmonic series, so you may be tempted to assume that this is always the case. It isn't.

Consider Figure 1 (see right). Imagine that the dashed black line in the diagram is a guitar string stretched between the nut and the bridge. You now use your fingertip or a plectrum to stretch the string a short way from its rest position, pulling it at its exact centre so that it becomes the red line in the diagram. In scientific terms, you have displaced the string at every point along its length, although at this instant it has zero velocity. Then you release it.

Each point on the string now starts moving towards the dashed line. Once the string reaches this point, its tension stops accelerating it, and begins to decelerate it as it stretches in the other direction. At some point soon after, the string comes to rest in the position shown in Figure 2 (above right), and at that moment the forces begin to pull it back towards its starting position.

It's tempting to think that a string vibrating in this fashion maintains its triangular shape throughout the cycle, but this is not so. Figures 3(a) and 3(b) (see right), show how two waves — one travelling left to right, the other right to left — combine to produce the wave motion of the string. As you can see, the string loses its triangular shape, and becomes a rapidly flattening trapezoid. It passes through the rest position as a straight line, becomes a trapezoid of opposite polarity, and then re-assumes its triangular shape at the opposite extreme before the forces pull it back again.

But what is the harmonic content of the

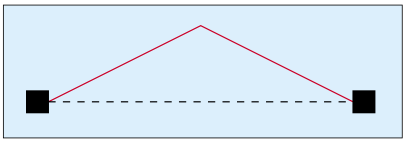


Figure 1: A guitar string plucked at its centre point.

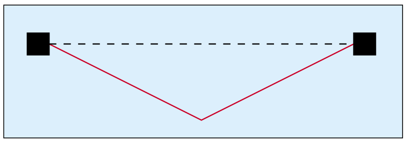
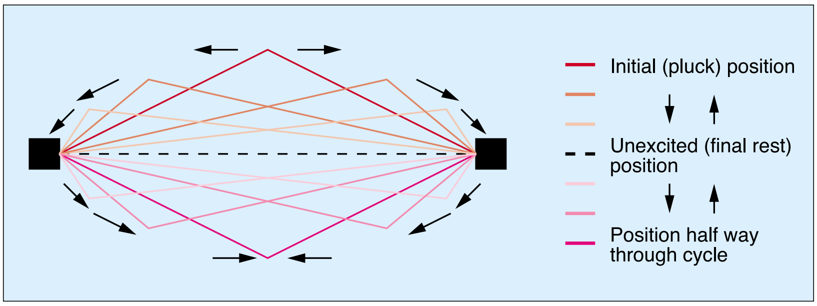
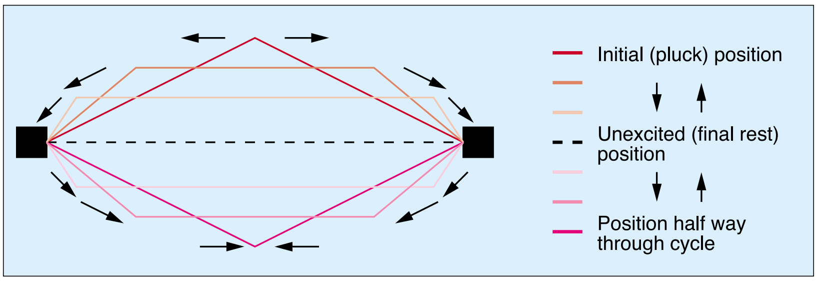


Figure 2: The plucked string a fraction of a second after release.



Initial (pluck) position
Unexcited (final rest) position
Position half way through cycle

Figure 3(a): How two pulses moving in opposite directions produce the wave motion of the plucked string.



Initial (pluck) position
Unexcited (final rest) position
Position half way through cycle

Figure 3(b): How the shape of the plucked string changes over time.

waveform produced by this motion? It's not trivial to perform a harmonic analysis to determine this, but don't worry; I'm not going to do the maths here. However, it's easy to visualise the result, because the starting point is a shape you know well — it's a triangle. This makes things simpler, as you will see.

Imagine that you have at your fingertips an additive synthesizer capable of producing numerous sine waves at the pitches and amplitudes of your choice. Let's suppose that you program it so that your oscillator produces a sine wave of frequency F, amplitude A, and starting phase 0º. Now add a second oscillator, with frequency 3F, amplitude A/9, and a starting phase of 180º. Now add a third, with frequency 5F, amplitude A/25, and phase 360º (which is the same as 0º)... and so on.

You will notice that each oscillator is producing the next *odd* harmonic in the series (or nA, if you like, where 'n' is any odd-number integer from 1 upwards), with an amplitude of $(1/n)^2$ and an initial phase shifted by 180º compared with the previous. Using a simple program such as Microsoft *Excel*, it's straightforward to show that the result of adding these oscillators' outputs is... a triangle wave (see Figure 4, above right).

Now, if the odd harmonics shown in Figure 4 conspire to create the triangle waveform shown, you might conclude that the component frequencies produced by a string plucked at its mid-point are also those of a triangle wave. Again, the proof of this is not trivial, but it makes an intuitive kind of sense. After all, if there were any *even* harmonics in the signal, there would have to be zero displacement at the centre of the string. You can see this in part 1 of Synth Secrets in *SOS* May 1999 (www.sound-on-sound.com/sos/may99/articles/synthsec.htm), where it is clear from the diagrams that all even harmonics on a vibrating string fixed at both ends have a 'node', or zero-displacement point, at the middle. The presence of such a node in the example in Figure 1 this month, however, is clearly impossible; the centre cannot be at zero, as it's the point at which the string is being plucked (see Figure 5, right).

So there it is... a guitar string — which is capable of vibrating at all the frequencies of its harmonic series — does not necessarily do so all of the time. When plucked at its centre, the string initially produces a triangle wave, and this has only *odd* harmonics.

Now let's return to this idea of nodes. If every second harmonic is missing when you pluck a string at its centre, it follows that:

- If you pluck the string a third of the way between its anchor points, every third harmonic will be missing;

- If you pluck the string a quarter of the way between its anchor points, every fourth harmonic will be missing...

...and so on. Again, this is because you can't have a node at the point at which the string is plucked.

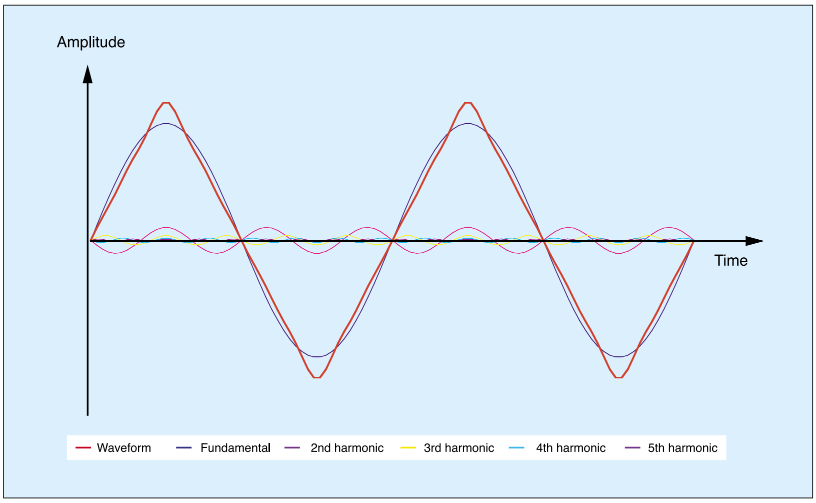


**Figure 4: The harmonic structure of a triangle wave.**

If you have access to a guitar, you can demonstrate this by moving your picking position up and down a string while holding the same fret. The result is the distinctive flanging sound produced by moving the 'holes' in the harmonic spectrum up and down the frequency spectrum. The generating mechanism may be different, but you're creating the same effect as a swept comb filter.

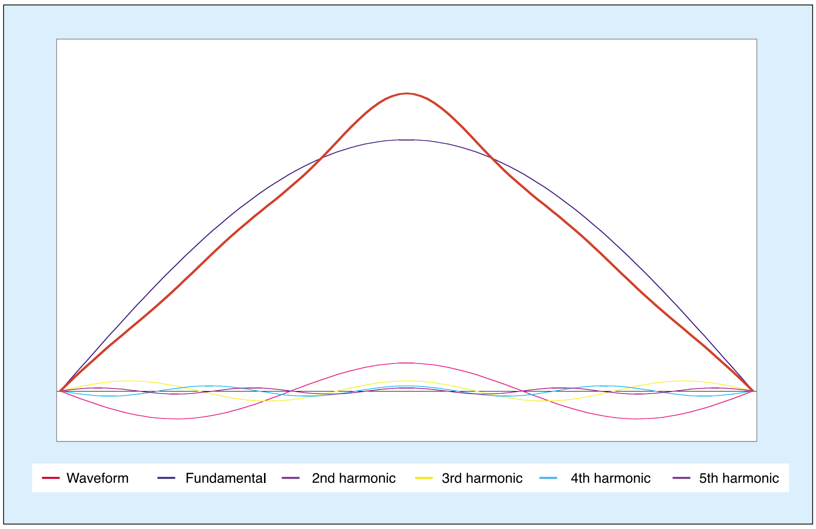


**Figure 5: The displacement components of a string plucked at its centre.**

So here's the first significant problem you encounter when trying to create a convincing guitar patch: although you might think of the guitar string as a simple oscillator, the plucking position determines its initial waveform and, therefore, its initial harmonic spectrum.

However, there are plenty of other complicating factors. Everything I've discussed so far has assumed a perfect triangular starting point for the string's vibrations. This never happens, because neither your fingertips nor a plectrum are infinitely small and hard. Consequently, the string will start vibrating with a rounded profile at the plucking point. This acts as a low-pass filter, suppressing the higher harmonics. Although you might think that this has an unimportant, or at best marginal effect, you only have to strum an acoustic guitar with something wide and soft (like your thumb) and then ▶

compare the sound to that achieved by strumming with something thin and hard, like the tip of a screwdriver, to see that this is not the case.

## Part 2: The Soundboard

Let's now move on from strings and consider another important component of the acoustic guitar... the soundboard formed by the upper surface of the instrument. Indeed, because this is a geometrically complex surface with a large hole in the middle, let's start by simplifying matters, and consider the properties of a flat, rectangular plate clamped on all four sides.

Like the circular membranes I described in part two of this series (see *SOS* June 1999 or www.sospubs.co.uk/sos/jun99/articles/synthsecrets.htm), rectangular plates have two dimensions. This means that they can vibrate in an East/West direction and in a North/South direction (actually, rectangular plates can also vibrate in a circular fashion, but I'm not going to consider this here for reasons of space). This means that I can't talk about single modes of vibration where plates are concerned (n = 1, 2, 3, 4... and so on) but must consider instead (m,n) modes of vibration where, for the sake of argument, 'm' is the number of nodes in the plate in the East/West direction, and 'n' is the number of nodes in the plate in the North/South direction.

This leads to the representations for the first handful of plate modes which you can see on the right of this page in Figures 6(a) to 6(f), where a shaded area is at any given moment 'up' and a white area is 'down' — or the other way around.

At this point, it's tempting to think that the plate is acting like a two-dimensional string and, to some extent, that's correct. The vibrations in the two dimensions are 'orthogonal' which means that the East/West waves are independent of the North/South waves, and the two do not interact. It is then tempting to think that the plate is producing two independent harmonic series of the sort produced by the string. After all, Figures 6(a) to 6(f) show that the vibrations in each direction are analogous to those in the string. Unfortunately, this is where the intuitive approach falls apart, and you need an understanding of wave mechanics to determine the vibration frequencies in each of the m,n modes.

If you do the maths, you find that the frequency produced by each mode depends upon size of the plate and the relative dimensions of its edges. For example, if the frequency of the 1,1 (fundamental) mode of vibration of the plate in Figure 6 were 100Hz, the 2,1 and 1,2 modes would have frequencies of approximately 277Hz and 171Hz. These do not fit any harmonic series, which is why a rectangular plate goes 'boing', rather than producing a note that sounds musical to our ears.

Now, let's return to the acoustic guitar. This has a top surface that is more complex than a rectangular plate, so its modes of vibration are more complex. Furthermore, the surface is supported by an intricate pattern of braces, and the shape and rigidity of these will complicate matters. This means that it's almost impossible to calculate the resonant modes of a guitar's top plate, although, using a sophisticated optical technique called interferometry, you can see them form dense patterns on the surface of the plate.

I have shown the very simplest of these modes in Figures 7(a) to 7(d), the series of diagrams at the bottom of this page, and it's not hard to see how they correspond to the first few vibrations of the simple plate. However, should the pattern of the braces change, these diagrams will be modified significantly, so please don't take them too literally.

## Part 3: Coupling The String & The Plate

To further develop your understanding of the sound of the guitar, you now have to consider what happens when you join the string to the top plate. Considered at its simplest level, this is what's called 'a system of couple vibrators', each with its own preferred modes of vibration. Of course, these are not free to oscillate in isolation (hence the word 'coupled') and each affects the other in complex ways. What's more, this coupling is not perfect, because the vibrations of the string are transmitted through a bridge that also responds to different frequencies in different ways. Ignoring the details of these interactions, it's possible to discuss the consequences of this coupling in qualitative terms.

Once you've completed the plucking motion, the plate absorbs energy from the string, thus sucking energy out of some of its modes of vibration. The plate then begins to vibrate at its own preferred frequencies. The vibrating plate then passes some energy back, exciting new modes in the string itself — including modes that were not present in the original vibration. This means that, within a cycle or two, the triangular waveform of the string changes to a new shape. But this isn't the end of the matter, because the modified vibrations in the string now excite the plate in a new way. The plate then responds differently, exciting new modes, and affecting the string in yet another way... and so on, until all the energy in the system radiates away as sound waves.

If this seems too convoluted to analyse, it isn't. However, it is hellishly complex. In synthesis terms, you have an *oscillator* (the string) whose output passes through a *resonator* (the plate), the response of which affects the harmonic content of the oscillator itself. In principle, there's no reason why

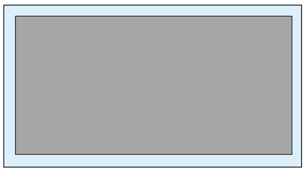


Figure 6(a): A rectangular plate vibrating in 0,0 mode (the fundamental).

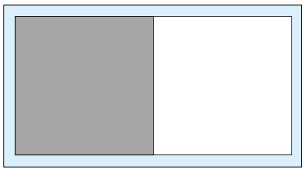


Figure 6(b): The plate vibrating in 1,0 mode.

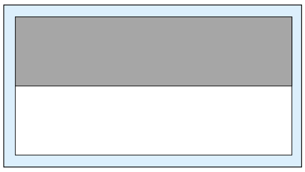


Figure 6(c): The plate vibrating in 0,1 mode.

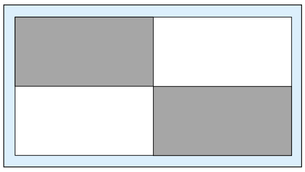


Figure 6(d): The plate vibrating in 1,1 mode.

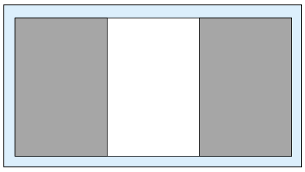


Figure 6(e): The plate vibrating in 2,0 mode.

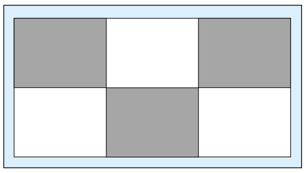


Figure 6(f): The plate vibrating in 2,1 mode.

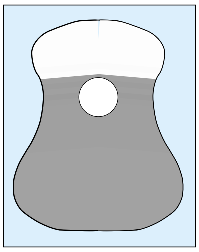


Figure 7(a): The 0,0 mode of a guitar plate.

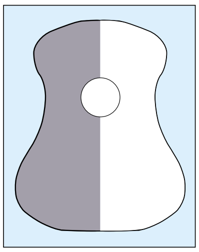


Figure 7(b): The plate's 1,0 mode.

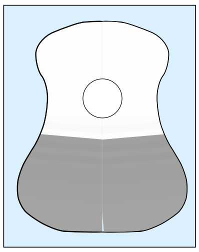


Figure 7(c): The plate's 0,1 mode.

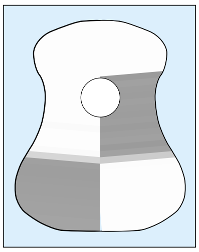


Figure 7(d): The plate's 1,1 mode.

▶ you shouldn't build a synth that does the same. But in practice... well, that's another matter.

## Part 4: The Hollow Body

Now you have to consider what happens when you add the sides and base of the guitar to the top plate, thus producing an air cavity within a hollow body.

You should recall from part 22 of Synth Secrets (see *SOS* February 2001 or www.sound-on-sound.com/sos/feb01/articles/synthsecrets.htm) that a hollow body will resonate at certain frequencies determined by its size and the shape of the cavity. However, the guitar is not as simple as the idealised room. This is because its body is not a perfect, rigid enclosure, and its vibrations are being 'driven' by the vibrations of the string and the top plate.

If, to simplify matters, I first treat the sides and back of the guitar as perfect, rigid enclosures, theory predicts that the air within the guitar body will have a 'comb' response at low frequencies. This is not unreasonable; at some frequencies, the air will flow out of the sound-hole in phase with the inward movement of the top plate, whereas at others, the two will be out of phase. This behaviour is analogous to that of a bass reflex loudspeaker.

However, guitars do not have perfect, immobile sides and backs, so a full analysis of the sound requires that I consider the vibration in each of these, too. The result of this shows that the back of the guitar introduces yet more resonances, as shown in Figure 8 (above right).

At this point, you might think that the guitar is complex enough, but I haven't even neared the end of my analysis. Consider this: when you pluck a string in a direction that is *perpendicular* to the surface of the top plate, the forces transmitted through the bridge are trying to bend the body along its length. However, when you pluck the same string in a direction *parallel* to the top plate, the string is trying to distort the body from side to side as well as bend it. If you look back to Figures 7(a) to 7(d), it should be obvious that the modes in Figures (a) and (c) are most likely to be excited by a perpendicular plucking action, whereas those in (b) and (d) will require some sideways component.

But what are the chances that you pluck a string in these precise directions? Virtually non-existent — so you have up-down and side-to-side components in every note, each exciting different body resonances in different proportions.

## Part 5: Amplitude Response

Everything I've discussed so far has described the mechanisms by which the guitar string generates the initial sound of the acoustic guitar, and the ways in which the body modifies the harmonic content of the waveforms thus produced. Until now, I've omitted any consideration of how the amplitude of the sound changes over time.

Perhaps surprisingly, it is possible to reduce the amplitude response of the acoustic guitar to a couple of simple generalisations, as follows: if you pluck a string parallel to the top plate, the amplitude of the resulting sound decays relatively slowly. If you pluck the same string perpendicular to the plate, the initial level is greater, but the sound decays more quickly. These can be depicted in simplified form as the amplitude envelopes shown in Figures 9(a) and 9(b).

Of course, you will rarely — if ever — pluck the string in exactly these fashions, so the true amplitude response will look more like that shown in Figure 10 (shown right). For any given initial displacement and plucking position, the height of the initial peak and the length of the final decay will then depend upon the angle at which you pluck.

## Part 6: Other Factors

I wish I could say that this is all there is to the sound of an acoustic guitar, but I can't. Many other factors influence what you hear. For example, I haven't mentioned the sympathetic string resonances that occur when more than one string is free to vibrate at any given moment. The importance of this is something you can demonstrate for yourself. Find an acoustic guitar and damp five of the strings. Then pluck the free one, listening to the tone of the resulting note. Now release the five damped strings, and play the same note on the sixth. It's different, isn't it? The reason for this is simple: after a few cycles, each of the undamped strings will be oscillating at the modal frequencies it shares with the plucked string. This is because the vibration of the plucked string passes through the nut and the bridge to the other five, exciting vibrations in each of these.

However, you now have nine vibrating resonators (six strings, the top plate, the bottom plate and the air in the cavity) rather than four, so the body resonances are excited in different ways, with different amplitudes. Thinking of this in terms of synthesis, you could say that the introduction of the second oscillator changes the interaction of the first oscillator and the resonator. The introduction of the third changes the interactions of the first and the second and the resonator... and so on. Ouch!

Another problem lies in the fact that the strings are not perfect harmonic oscillators. This is because they do not form perfect angles at the nut or the bridge. The finite cross-section of the string ensures that it curves at these points, thus making

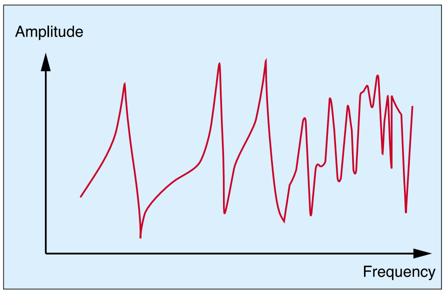


Figure 8: A representation of the low- and mid-frequency response of an acoustic guitar body.

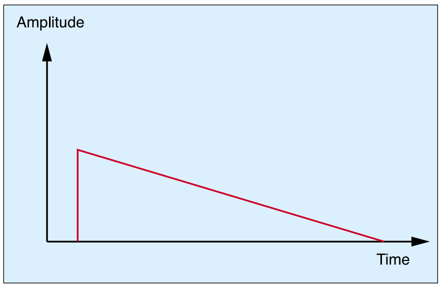


Figure 9(a): The amplitude envelope of a string plucked parallel to the guitar's top plate.

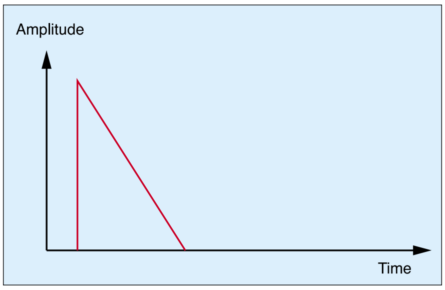


Figure 9(b): The amplitude envelope of a string plucked perpendicular to the top plate.

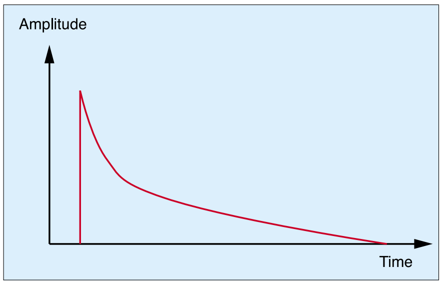


Figure 10: A realistic decay curve for a plucked guitar note.

its active length slightly shorter than the idealised view would suggest. The degree to which this happens depends upon the wavelength and amplitude of the vibration, so the string appears shorter at high frequencies and high amplitudes than it does at low frequencies and low amplitudes. This sharpens higher, louder harmonics, further complicating any analysis of the string's harmonic spectrum, as well as the guitar body's response to it.

Finally, you have to consider the radiated sound pattern of the acoustic guitar. Like all acoustic instruments, the frequency response and the harmonic content of notes differ from one listening position to another. Experiments show that the loudness of some frequencies can differ by as much as 20dB if you shift your listening position. Likewise, if guitarists shift their seating position, or the angle of their guitar, the same dramatic change can occur.

## Part 7: Synthesizing The Acoustic Guitar?

At this point, you can start to consider how to patch an analogue synthesizer in order to produce an acoustic guitar sound. But if you don't know where to start, don't worry; neither does anybody else. Let's look at the problems:

- Each string produces a different waveform depending upon the plucking position;

- The shape and hardness of your fingers or the plectrum influences the high-frequency content of the initial waveform;

- The amplitude envelope of the oscillators depends upon the direction in which you pluck the string(s);

- The strings' harmonics are 'stretched' as the pitch increases and/or the excitation increases in amplitude;

- The six strings interact with each other in different ways, depending upon their pitches and the number of them which are free to vibrate at any given time;

- Each string interacts with a system of complex resonators (the guitar body) that absorbs energy and then directs it back to all the strings, exciting harmonics that may not be present in the initial waveform;

- The body has many densely packed resonances and anti-resonances that can not be imitated using conventional equalisers or filters;

- The nature of the resultant sound is extremely dependent on the position of the listener and the angle between the listener and the instrument.

There are other factors, but these eight give you a good idea why you can not create authentic-sounding acoustic guitar patches using analogue subtractive synthesis. This is one occasion when only digital technology will do! SOS

# synth secrets

I ended last month's explanation of the complexities of attempting to synthesize guitars with this statement: "You cannot create authentic-sounding acoustic guitar patches using analogue subtractive synthesis. This is one occasion when only digital technology will do!"

Since I'm writing this before that instalment of Synth Secrets is published, I have no idea whether this assertion has stirred up a storm of protest, or whether you have simply accepted it. Nonetheless, it's not very sporting to walk away from an argument (real or imagined), so I thought that it would be fun to try to patch an analogue model of the guitar sound, and to see how far it falls short of the ideal.

### A Real Guitar

If we take a very simplistic view of an acoustic guitar, it seems straightforward to create a synthesizer patch that will imitate its major characteristics. Firstly, the instrument has six strings, so the synthesizer should have six voices. Secondly, strings produce complete harmonic series, so we set each of the oscillators in the voices to produce a sawtooth wave.

Next, we apply two bits of acoustic knowledge that are so familiar that they are almost truisms. The first of these is: the higher the pitch of an acoustic sound, the brighter it will be. The second is: at any given moment, the louder the sound is, the brighter it will be.

Recreating these characteristics on a keyboard synthesizer is easy: we use a low-pass filter, the cutoff frequency of which responds to the pitch CV (thus taking care of point one) and to a contour generator that duplicates the loudness characteristic of the sound (to take care of point two). The easiest way to do this is to use the same contour generator for both the brightness and the loudness. Since we are considering the case where the guitar's strings are plucked, we can model each string's amplitude response using an unconditional AD contour generator and a VCA. ('Unconditional', in this context, means that the Decay stage of the contour is completed whether or not we release the key producing the note.)

Putting it all together produces the simple six-voice synthesizer described by Figure 1. Unfortunately, if you set up this patch on a synth with the appropriate architecture (the Roland Juno 6 is a perfect example), you'll find that, no matter what you do with the parameters, the sound remains singularly unlike a real guitar.

## PART 29: THE THEORETICAL ACOUSTIC GUITAR PATCH

Having explained last month the reasons why analogue synthesis of guitar sounds should be well-nigh impossible, **Gordon Reid** puts the theory to the test...

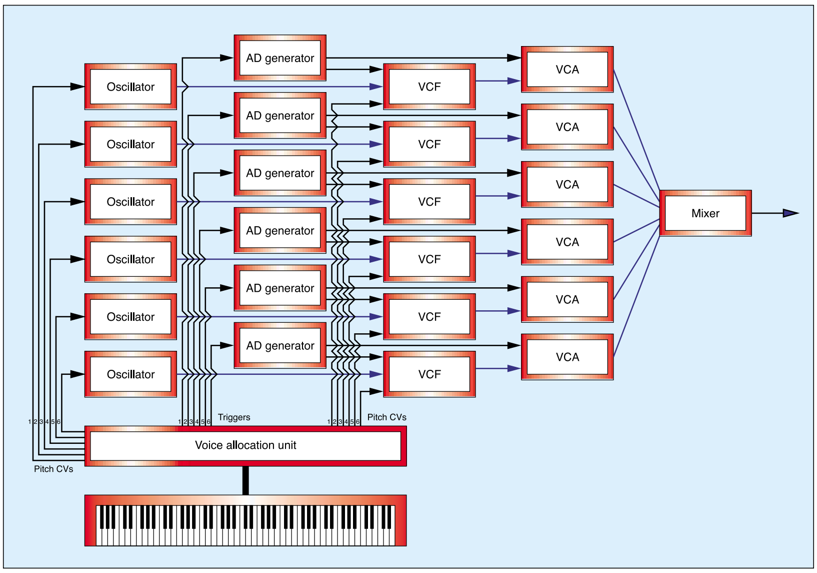


Figure 1: A simple six-oscillator synthesizer.

There are two reasons for this. The first lies in the voice assignment — the way in which the synthesizer allocates its voices to the notes you play on the keyboard. The second lies in the inadequacies of the sound itself.

### Voice Assignment

Let's start by considering the way in which a guitar allocates notes to its 'oscillators'. Consider the sequence in Figure 2. This shows a simple chromatic scale running from G# to B.

If you play this on the 'G' string alone, you will start on the first fret (G#), thereafter stepping up each of the next three frets to play A, A# and B. This has an important acoustic consequence: the plucking of each new note terminates the previous one, reinitialising the brightness and loudness contours (see Figure 3, above right).

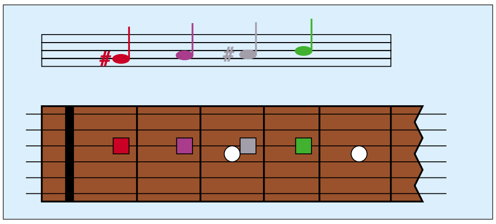


Figure 2: A four-note sequence played on one string of an acoustic guitar.

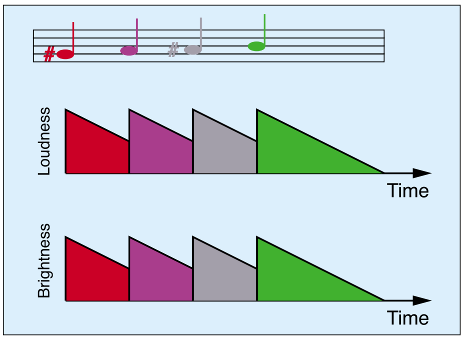


Figure 3: The brightness and loudness contours of the notes in Figure 2.

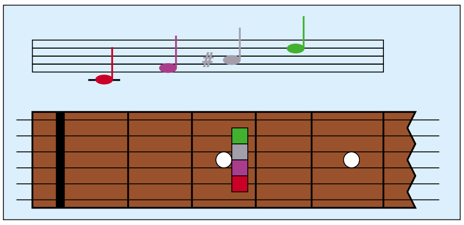


Figure 4: A four-note sequence played on four strings of an acoustic guitar.

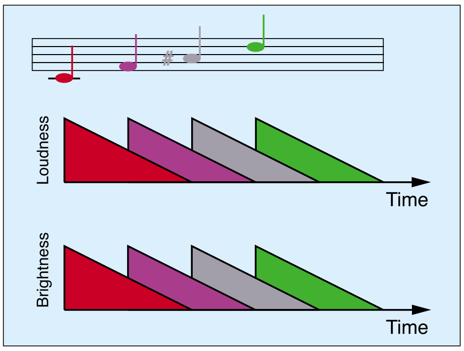


Figure 5: The brightness and loudness contours of the notes in Figure 4.

Now let's consider the arpeggio in Figure 4. This shows the four notes C, F, A# and D played across the A, D, G and B strings. In this instance, there's nothing stopping each note from dying away slowly, with its brightness and loudness decaying naturally to silence (see Figure 5).

So here's our first difficulty: how do we decide whether any given note in our guitar imitation should curtail a previous one and, if so, which one? This is a problem that needs a computer for its solution. No matter — all but a tiny handful of polyphonic synthesizers use digital keyboard scanners, so this should not prove to be an impediment, provided that someone can work out a suitable algorithm that analyses the notes and causes them to respond appropriately.

There are two products I know that are capable of this. One is the Korg KARMA; the other is the Charlie Lab Digitar, a superb little device that takes previously sequenced MIDI notes, lets you strum/pick a guitar-like controller, and works out the correct virtual string assignments in real time. It then produces a new MIDI stream that imitates the playing of a real guitar. Not only does it create the correct MIDI Note *Ons*, it also outputs MIDI Note *Offs* to truncate notes that would be curtailed when a new note was played on the same string. It even calculates the inversions up and down the neck for you! If you point the output from the Digitar at a suitable set of acoustic guitar multisamples, the results can be remarkably realistic.

The Charlie Lab Digitar (see *SOS* January 1995 or www.sound-on-sound.com/sos/1995_articles/jan95/charlielab.html) can simulate the way a guitar plays chords.

But what has this to do with imitating the acoustic guitar using subtractive synthesis? Quite a lot, actually, because there's nothing stopping you from directing the output from the Digitar to any analogue synth with a MIDI input and internal MIDI/CV converters. So now we need only create the perfect guitar sound, and we're done...

## The Sound

I ended Synth Secrets 28 with eight reasons why it is hard, if not impossible, to produce a realistic acoustic guitar sound using subtractive synthesis. Let's take a look at each of these, and see whether we can overcome the problems they represent:

*Each string produces a different waveform depending upon the plucking position.*

Last month we saw how, when plucked in its centre, a guitar string produces a triangle wave for the first few cycles. I then explained that the interaction of the string and the soundboard excites other modes of vibration, including the even harmonics that are missing from the initial waveform.

▶

▶ So which synthesizer waveform should we use for our sound? One answer remains the sawtooth, the only common waveform that includes all the harmonics in the series. It's inevitable that these harmonics will be present in the wrong proportions, but at least they're present. However, a better choice would be to use an oscillator with a CV input that allows us to modify the harmonic content of the wave during the course of the note. At this stage, we've no way of knowing what we might apply as the modifying CV, but at least the facility is available (see Figure 6, above right).

*The shape and hardness of the fingers or the plectrum influences the high-frequency content of the initial waveform.*

This suggests that we need to add an equaliser to the signal path before the signal passes through any other modifiers. The simplest that might be suitable is a high-frequency 'shelving' equaliser, similar to the 'High' EQ on a small mixing desk. With the high frequencies boosted, our patch will, in effect, model a hard plectrum. With the high frequencies suppressed, the sound will veer towards that produced by a soft plectrum or finger-picking with the pad of the fingertip (see Figure 7).

*The amplitude envelope of the oscillators depends upon the direction in which you pluck the string(s).*

As stated last month, a string plucked parallel to the top plate of the guitar has a lower amplitude, but a longer decay than one plucked perpendicular to the plate. Fortunately, it's easy to model this. We use an AD contour generator that offers simultaneous control over the amplitude of the Attack (AL — Attack Level), as well as the Decay Time (DT). If the strum is parallel, the CV causes AL to decrease and DT to increase. If it is perpendicular, AL increases and DT decreases. We can also use this part of the patch to model the strumming/picking intensity. We do this by directing the velocity CV from the controller keyboard to the Attack Level and Decay Time, increasing each with greater velocity (see Figure 8 above).

*Strings' harmonics are 'stretched' as the pitch increases and/or the excitation increases in amplitude.*

This is a tricky one, and there's nothing we can do to model it using subtractive synthesis. After all, the harmonics of a conventional oscillator lie at exact multiples of the fundamental frequency. If the real string does not conform to this relationship... well, that's tough. So let's move right along to...

*The guitar body has many densely packed resonances and anti-resonances that cannot be imitated using conventional equalisers or filters.*

As we saw last month, a guitar's resonances are fiendishly complex and defy any attempts at simple analysis. Academic institutes spend considerable amounts of research time and funding on this problem, and this research has led in part to the algorithms now used in physical-modelling synthesizers. But as for a model suitable for implementation using analogue electronics, as far as I know, no such thing exists. Nevertheless, we could — with a lot of time and money — employ a large number of parametric EQs with very high 'Q' values to produce something that approximates a body response.

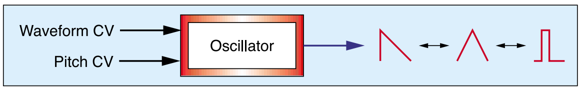


Figure 6: An oscillator capable of morphing from one waveform to another smoothly in real-time.

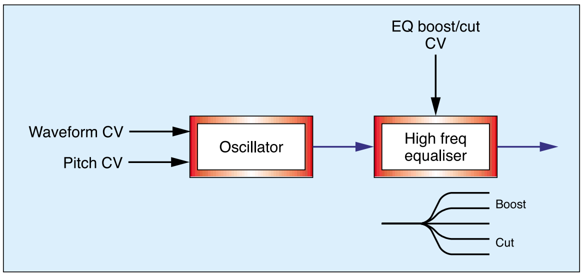


Figure 7: Modelling the nature of the pick.

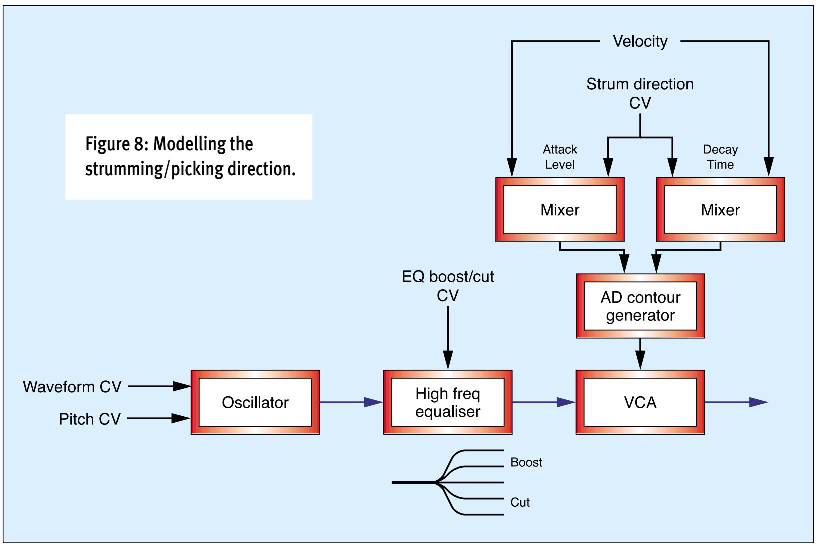


Figure 8: Modelling the strumming/picking direction.

Figure 9 shows the low- and mid-frequency body resonances of the guitar discussed last month, and Figure 10 shows an approximation of this, crafted via 12 extremely precise — and no doubt extremely expensive — parametric EQs. Graphic EQs are useless for this, because their bands are far too wide. Comb filters are useless because their peaks and troughs are in the wrong places.

As you can see, even in this idealised representation, which is far from the real response of even the most precise EQ bank, the responses do not look identical, and your ears will certainly be able to tell the difference. But it's a step in the right direction. Figure 11 shows a six-band parametric EQ bank added to the synthesizer patch. In all likelihood, we would need at least half a dozen of these to create a frequency response that even approached something meaningful. ▶

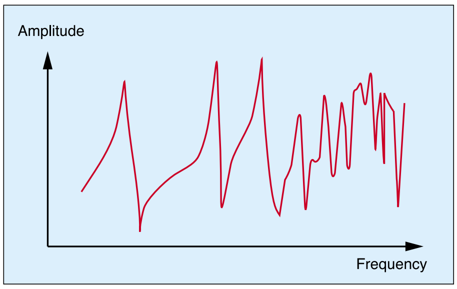


Figure 9: The measured low- and mid-frequency response of a guitar body, as first shown in Synth Secrets 28.

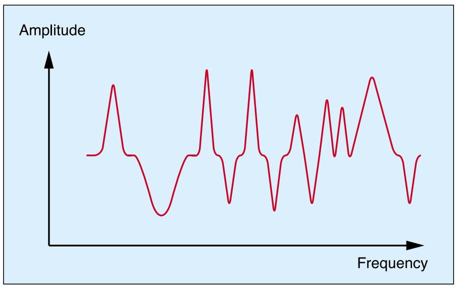


Figure 10: An idealised view of the response of 12 parametric EQs set up to respond like a guitar body.

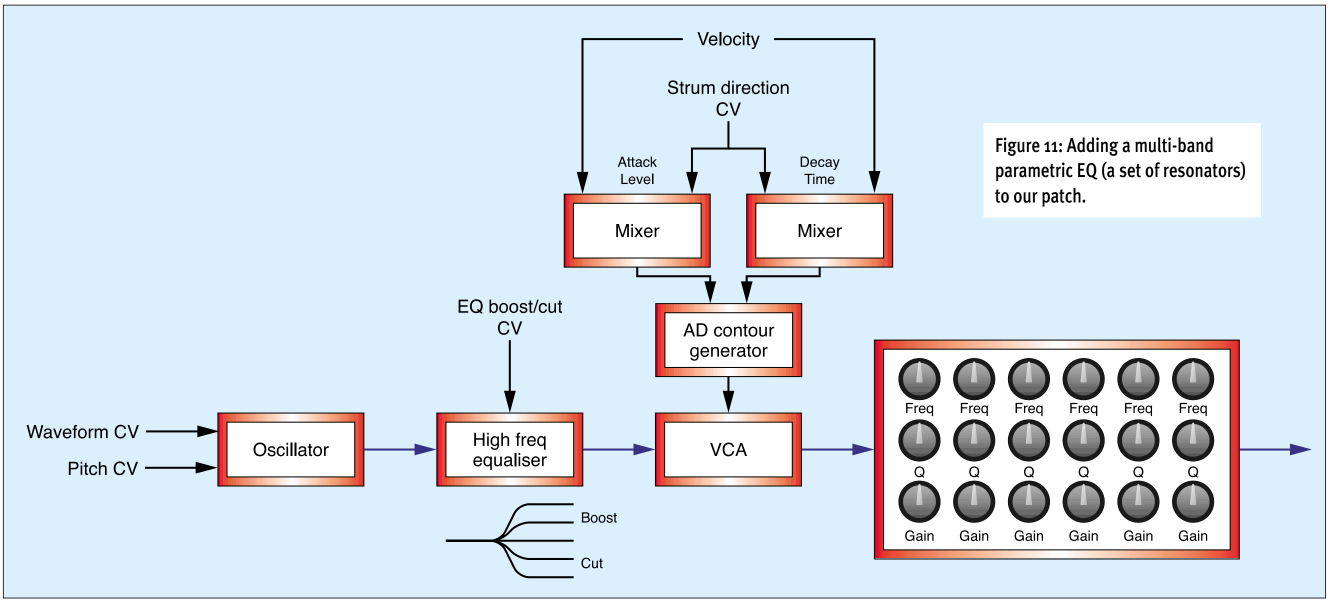


Figure 11: Adding a multi-band parametric EQ (a set of resonators) to our patch.

*Each string interacts with a system of complex resonators (the guitar body) that absorb energy and then direct it back to all the strings, exciting harmonics that may not be present in the initial waveform.*

At this point, we need to create a feedback loop that will take the output from the filter banks and use it, in some way, to modify the waveform produced by the oscillator. The simplest way to do this would be to route the resonators' output directly back to the waveform CV input, thus modifying the wave at the same frequency as the signal itself. Unfortunately, this means that some harmonics are being added to, and removed from, the signal at audio frequencies... which is amplitude modulation. This will result in the creation of unwanted side bands, and is therefore quite unsuitable for our purposes (see part 11 of Synth Secrets, in *SOS* March 2000, or surf to www.sound-on-sound.com/sos/mar00/articles/synthsecrets.htm). What we want is something that responds to the output from the EQ bank, but does so more slowly, making the waveform change more subtly over the course of a few cycles.

One way to do this would be to derive a side-chain from the audio signal, high-pass filter it to separate out the high-frequency content, and then pass the result through an envelope follower to determine the amplitude of the separated signal. If the envelope follower responds too rapidly, we can use an S&H generator and a slew generator to create a smoothly changing voltage that varies within the ▶

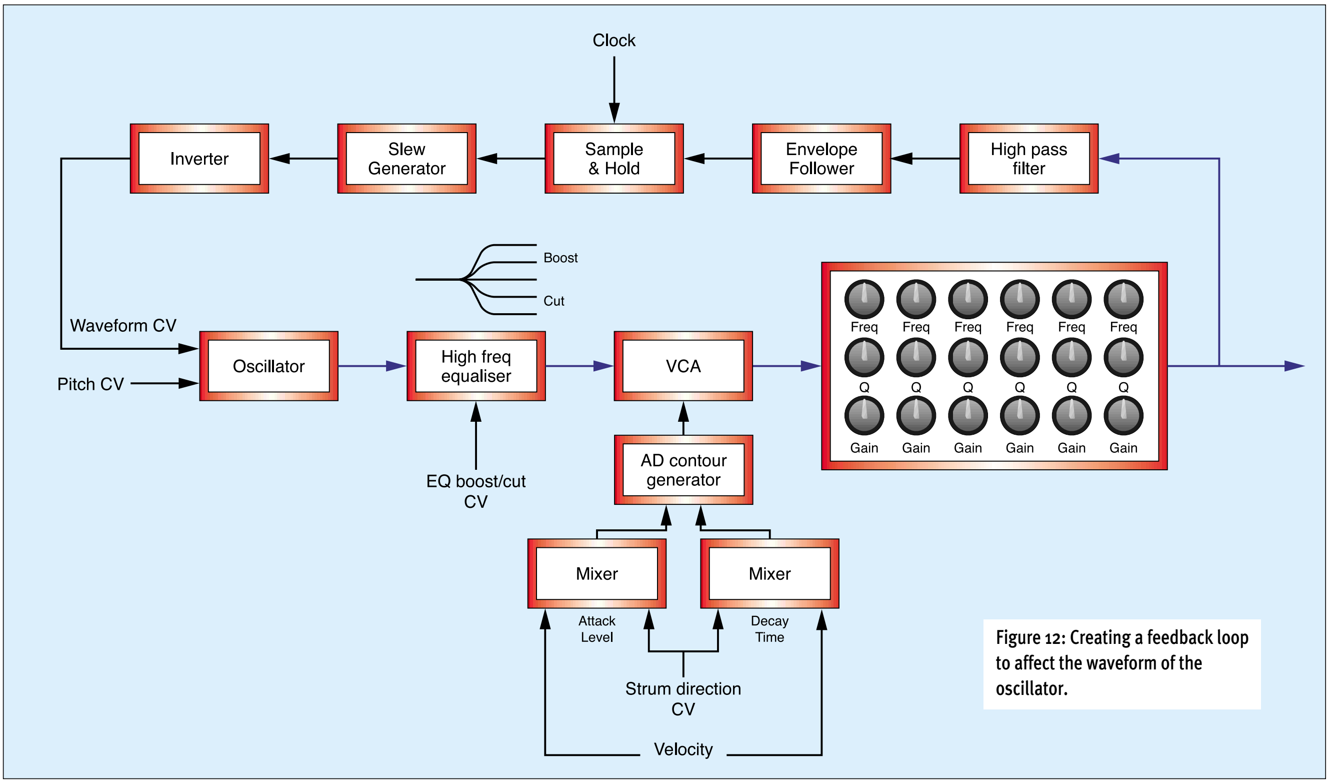


Figure 12: Creating a feedback loop to affect the waveform of the oscillator.

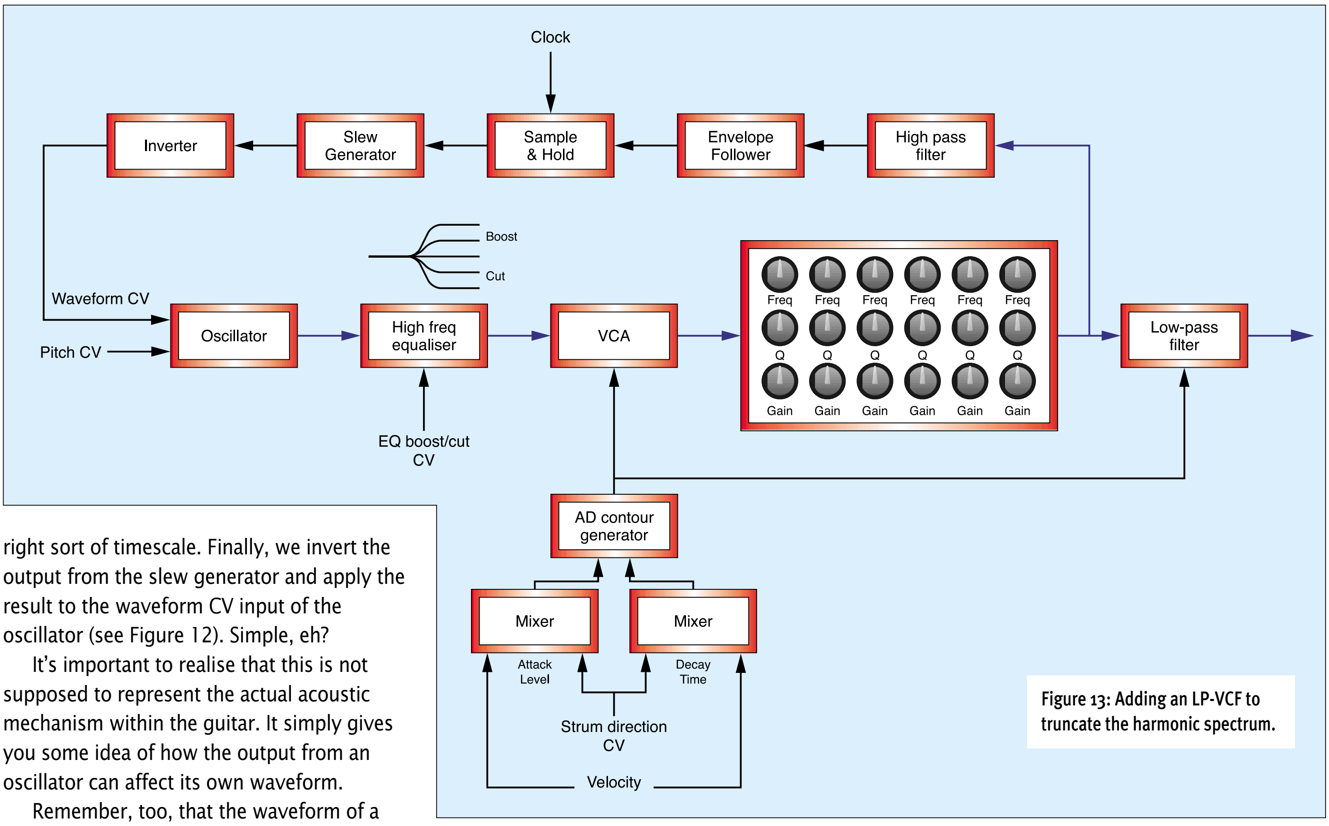


Figure 13: Adding an LP-VCF to truncate the harmonic spectrum.

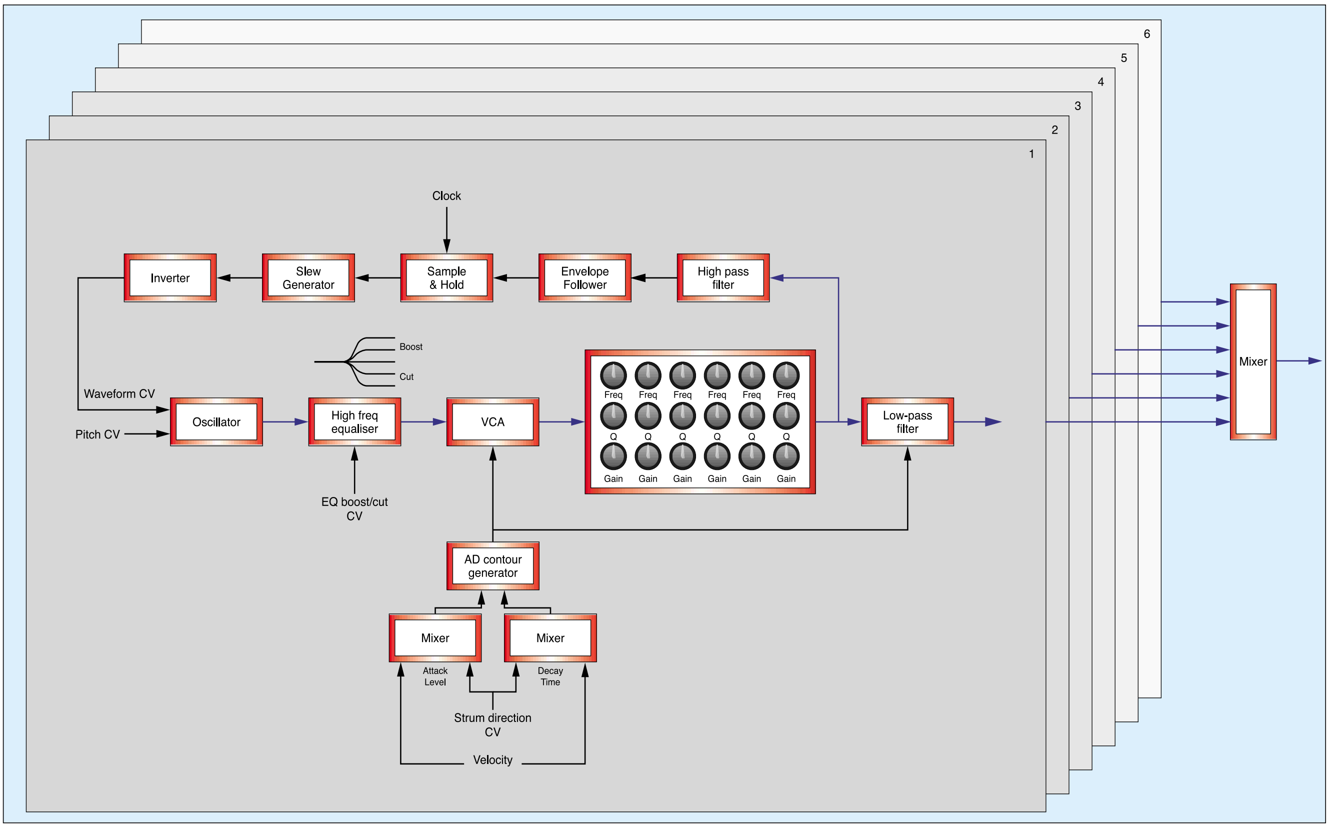
right sort of timescale. Finally, we invert the output from the slew generator and apply the result to the waveform CV input of the oscillator (see Figure 12). Simple, eh?

It's important to realise that this is not supposed to represent the actual acoustic mechanism within the guitar. It simply gives you some idea of how the output from an oscillator can affect its own waveform.

Remember, too, that the waveform of a real plucked string tends towards toward a sine wave as time passes, with nothing but the fundamental present as the oscillation decays to inaudibility. As in Figure 1, we can model this by using a low-pass VCF whose cutoff frequency is controlled by the AD contour generator that also controls the audio signal VCA (see Figure 13).

Oh heck! Every six-string guitar has six of the systems shown in Figure 13, and each of them interacts with the others, modifying the waveform of the other five, as well as changing the decay characteristics of each (one string can provide energy to another, thus altering how it decays over

Figure 14: Treating the six strings as independent entities.



time). These interactions are far beyond the scope of Synth Secrets, so our only sensible course of action is to treat each string as an isolated entity, as shown in Figure 14.

Having done so, we must remember to set up each of the virtual voices individually, because the initial tone and amplitude characteristics of, say, a 0.052-inch wound bottom 'E' string are quite different from a 0.009-inch top 'E' string.

*The nature of the resultant sound is determined by the listening position and the angle between the listener and the instrument.*

Good grief... we need *another* complex EQ that models the spatial projection of the guitar body. Unfortunately (again) there's no standard way to set this up, so we will simply have to experiment to find something that sounds natural and pleasing. The block diagram for this is shown in Figure 15 above.

## Guitar Performance Synthesis

So there you have it: a huge modular synthesizer, patched to recreate the sound of an acoustic guitar. But we've still ignored the stretching of the harmonics, and we've not even attempted to recreate the true feedback characteristics of the string/body interaction. So, at best, this sound will merely approximate a guitar. Oh yes — and we've ignored the fact that filter banks of this size and precision don't exist in the world of analogue synthesizers.

Nevertheless, let's ignore all of these problems, and assume that we've managed to create a set of authentic string tones. Unfortunately, we're still not going to convince any listener that we have a real guitar in our hands. This is because we have ignored all the other characteristics — vibrato, pitch-bends, slides, and string squeaks, among others — necessary to produce a convincing performance.

The first of these is not a big problem, because there are many ways of adding vibrato to an audio signal. The cheat's method would be to use a set of six LFOs (one for each string) with a set of six poly-pressure controlled VCAs. If this seems a little fanciful, I can think of one (mostly) analogue synth, the Sequential Prophet T8, that offers polyphonic aftertouch coupled to multiple LFOs. However, bear in mind that this mechanism only provides control over the depth of the vibrato, not the speed. I would prefer two mechanisms that could control both of these characteristics independently, but (like so much else in this article) that's wishful thinking. Perhaps a satisfactory compromise would be to divert the pressure CV to both the amplitude and the LFO speed, as shown in Figure 16 above.

We could use the same configuration to add pitch-bend to the strings, simply by taking a second output from the pressure CV and adding this to the pitch CV in the synth's Mixer. Unfortunately, this would mean that we could never separate pitch-bend from vibrato, and no guitarist could, or would, play like this. So we need another controller source. We can't use velocity — that's controlling the loudness and tone of the note. This leaves us with pitch wheels and foot pedals. However, we would need six of these to create a convincing performance, and that just isn't practical. Consequently, I think that we're going to have to ignore individual pitch-bends, settling for just a single bend applied to the whole patch. In keeping with conventional synthesizer architecture, we can use the pitch-bend wheel to apply this (see Figure 17). It will sound reasonable for one string, and maybe even when two are sounding, but woe betide you if you bend all six simultaneously. It will sound horrible.

Now, what about those string squeaks? It's easy to add them using a PCM guitar sound, whereby you add a sample of the squeak under every note, but restrict it to sounding within a limited range of MIDI velocities: say, 124 to 127. This means that you can play the sound normally within (almost) the full dynamic range of the keyboard, but can add a squeak by hitting the appropriate note as hard as you can. We can't do this using any reasonable analogue synth (it's possible, but far from practical), so we'll have to

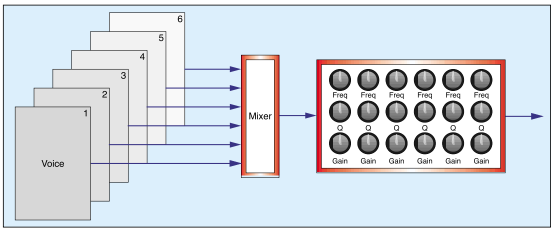


Figure 15: Shaping the sound to imitate changes in listening position.

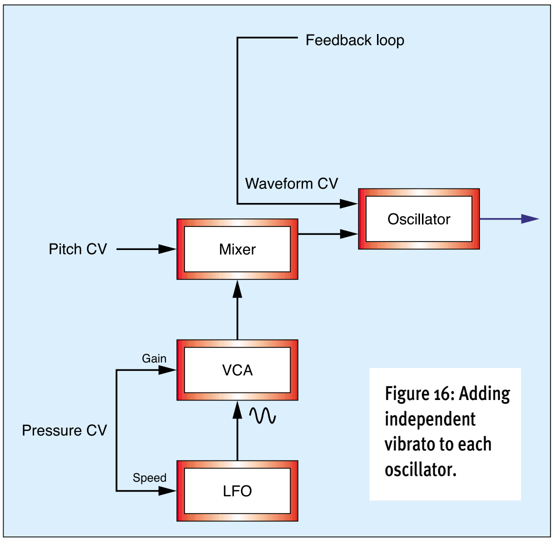


Figure 16: Adding independent vibrato to each oscillator.

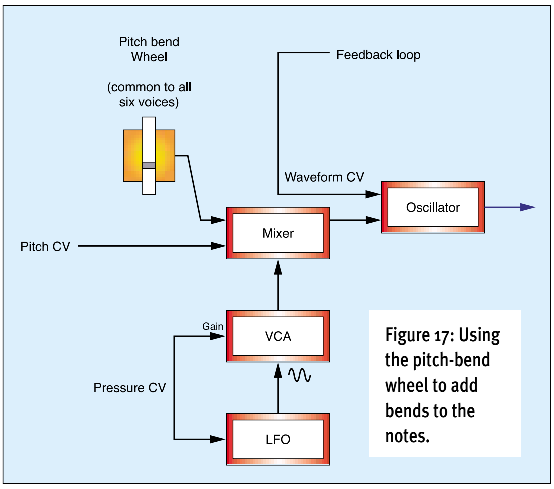


Figure 17: Using the pitch-bend wheel to add bends to the notes.

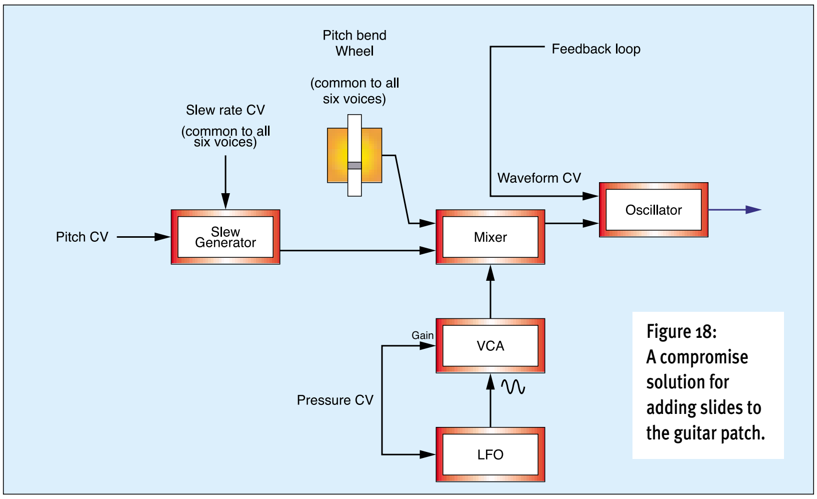


Figure 18: A compromise solution for adding slides to the guitar patch.

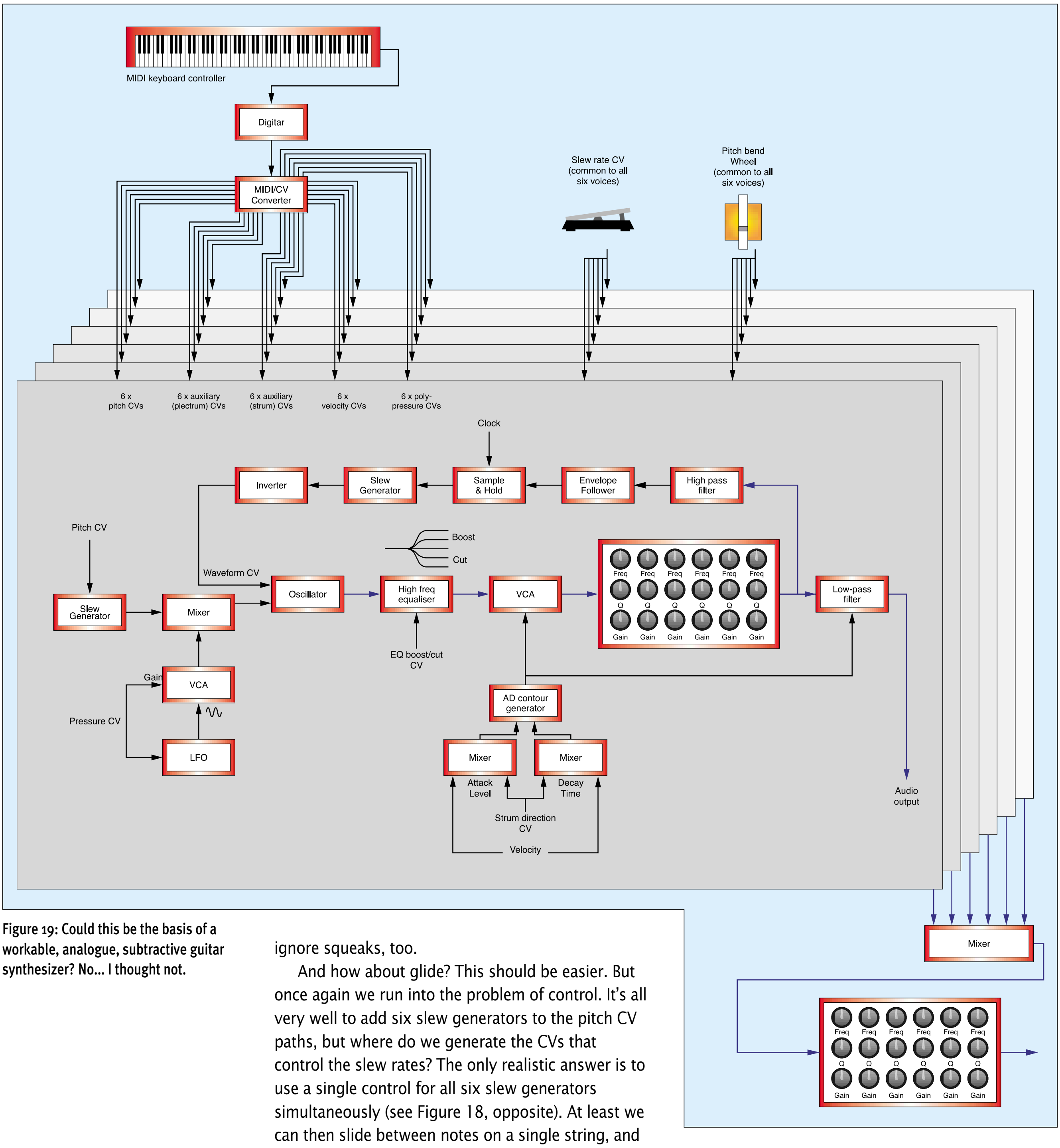


Figure 19: Could this be the basis of a workable, analogue, subtractive guitar synthesizer? No... I thought not.

ignore squeaks, too.

And how about glide? This should be easier. But once again we run into the problem of control. It's all very well to add six slew generators to the pitch CV paths, but where do we generate the CVs that control the slew rates? The only realistic answer is to use a single control for all six slew generators simultaneously (see Figure 18, opposite). At least we can then slide between notes on a single string, and slide whole chords up or down the neck. But as for single-note slides within a chord, forget it. Oh yes, and don't disregard the fact that a guitar is a fretted instrument, so we should be quantising the slides (which travel across the frets), but not the pitch-bends or vibrato (which don't). If you would like to give yourself a headache, try to work out a patch that will offer this. It's a superb example of how difficult it is to create and manipulate multiple control signals using conventional keyboard synthesizers.

## Conclusions

Let's finish by putting the whole thing together, in Figure 19, to see how it looks (see above). I make no claims for the diagram's veracity, nor do I state that it will do the job as advertised. But it will give you some idea of the complexity of the problem.

At this point, I suppose you would like to know how it sounds. Well, to be honest, I haven't a clue. Ignoring the stretched harmonics again, as well as the fact that the filter banks (resonators) don't exist, I simply don't have access to a poly-pressure sensitive, modular, analogue synth with well over 100 modules (plus the numerous signal splitters called multiples that I have ignored for the sake of clarity).

As I've said twice already, this is one occasion when only digital technology will do! SOS

*I would like to thank Christopher Hicks of CEDAR Audio for his invaluable advice during the writing of Synth Secrets Parts 28 and 29.*

# synth secrets

## PART 30: A FINAL ATTEMPT TO SYNTHESIZE GUITARS

**Having proved that subtractive synthesis of an acoustic guitar is completely impractical,** Gordon Reid **tries his hand at the electric variety, and deconstructs some past attempts to emulate the sound via analogue means.**

For the past two months, I've used about 7000 words and 40 diagrams to explain why it's not practical to emulate guitar sounds using traditional analogue synthesis. So why does every synthesizer patch book include voices with names such as 'Jazz Guitar', 'Funky Wah-Wah Guitar', 'E-Guitar', or something similar? The answer to this lies in the nature of the guitar that the patch is attempting to emulate.

In the last two parts of this series, I restricted my analysis to the qualities of the acoustic guitar, with all of its resonant surfaces and complex enclosed spaces. But let's ask what happens when we take the six primary oscillators of the acoustic guitar (the strings) and attach them to a *solid* body. What we now have is a guitar that comprises six strings, a neck, some form of tuning mechanism, and a wooden plank that acts as its resonator. We can add devices such as electromagnetic pickups, volume and tone circuitry, a tremolo system, and even digital signal processing technology — but none of these affect the basic sound produced by the instrument. Indeed, until you plug this 'electric' guitar into an amplifier and speaker, the vibrations of the strings themselves dominate its sound. Sure, the weight of the body and the quality of the material from which it is constructed will have a subtle effect, but there is no acoustic cavity and no soundboard to amplify and modify that sound.

Extending this line of reasoning, we could hypothesise (correctly) that the traditional guitar shape is no longer relevant to the sound of the electric guitar. It is for this reason that the glam-rock bands of the '70s could play guitars shaped like stars, the letter 'V', that were circular, or even carved in the shape of Africa (this tells us why they *could*, but gives us no inkling about why they *did*. I guess you just had to be there).

But why stop there? Since the electric guitar is insensitive to its body shape, it seems reasonable to speculate that it will be similarly indifferent to the material from which the body is made. And, although many guitarists will wax lyrical about their favourite lumps of dead tree, this is not such a stupid idea. There have long been graphite guitars and basses, and I remember wanting an aluminium-necked Kramer in the early '80s.

So, let's ask what this has to do with analogue synthesis. The answer is obvious: none of the guitar patches designed for early analogue synthesizers attempted to produce the complex sound of a nylon-strung, hollow-bodied acoustic instrument. All of them emulate, to a greater or lesser extent, the (hypothetically) simpler sound of the solid-bodied electric guitar.

### The Electric Guitar

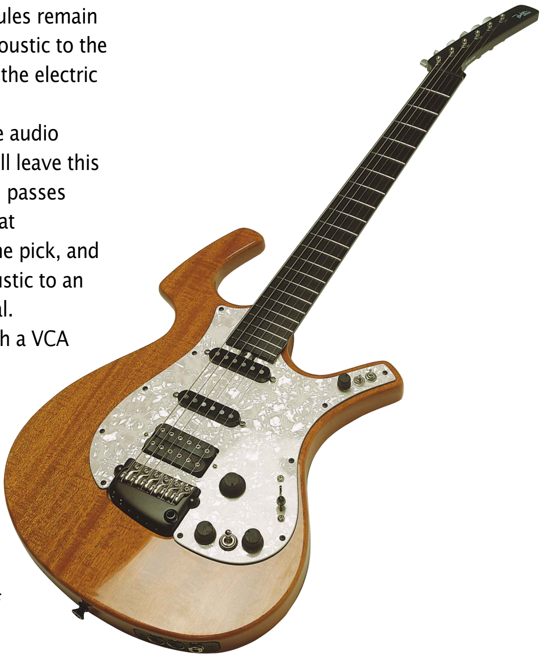
We're going to try to use our knowledge of the electric guitar to simplify — or, at the very least, make more practical — the analogue 'model' of the acoustic guitar that concluded last month's Synth Secrets (see Figure 1, opposite).

Starting on the left-hand side of this, we find the slew generator, LFO, VCA and mixer that provide much of the performance capabilities within the patch. Since all six-string guitars are equally capable of slides, bends and vibrato, these modules remain unaffected by the transition from the acoustic to the electric guitar, so we will retain them in the electric guitar patch.

The next element, and the first in the audio signal path, is the oscillator, and we shall leave this where we found it. The output from this passes through the high-frequency equaliser that I introduced to model the hardness of the pick, and this too survives the move from an acoustic to an electric guitar patch. So far... so identical.

Next, the audio signal passes through a VCA controlled by an AD contour generator. The loudness and brightness envelopes of an unamplified electric guitar are very different from those of an acoustic guitar but, although the speed and the precise shape of the decay differ, the underlying characteristics remain the same. Therefore, we will also leave this part of the patch untouched.

After this, the audio passes through a complex filter bank that imposes the resonant characteristics of the guitar body upon it. Hang on a moment... haven't I implied that the electric guitar's



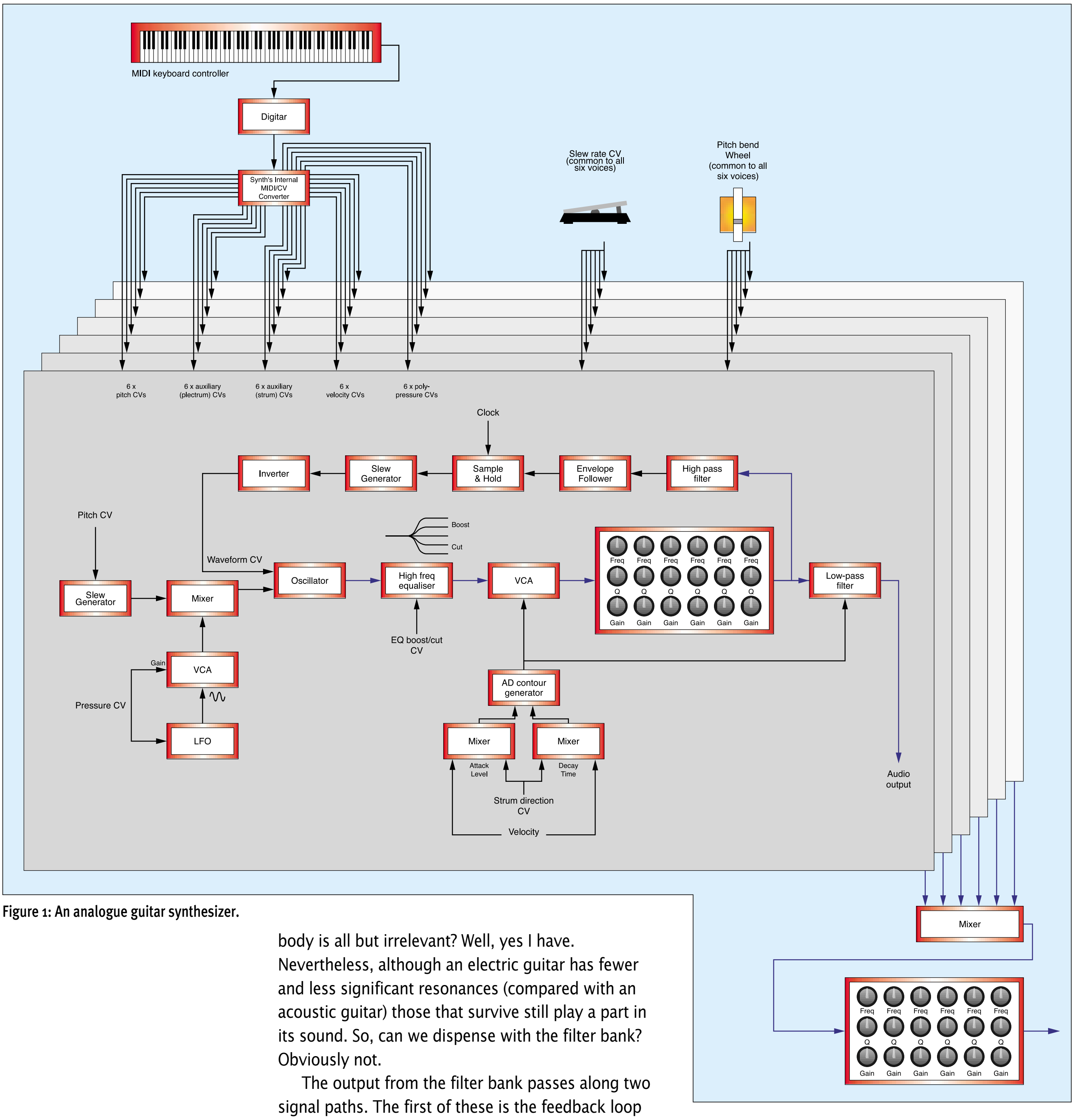


Figure 1: An analogue guitar synthesizer.

body is all but irrelevant? Well, yes I have. Nevertheless, although an electric guitar has fewer and less significant resonances (compared with an acoustic guitar) those that survive still play a part in its sound. So, can we dispense with the filter bank? Obviously not.

The output from the filter bank passes along two signal paths. The first of these is the feedback loop that emulates the way in which the body of the acoustic guitar absorbs energy from the vibrating strings and then passes it back in ways that modify their vibrations. Because an electric guitar's body absorbs and radiates little energy in comparison with the acoustic guitar's, it's tempting to suggest that we can dispense with this loop. After all, it would simplify the patch considerably. Unfortunately, we can't do so with a clear conscience. The interactions between the strings and the body are less important, but we would still be removing a part of the model's sound-generation system.

The second audio path exiting the filter bank passes through a low-pass filter that models the decreasing high-frequency content of the signal as the note decays in time. This too must stay, as must the mixer that combines the sounds produced by the six voices.

We now come to the filter bank that emulates the way in which the sound of the acoustic guitar depends upon the relative positions of the player and listener, and the relative angle at which the player holds the guitar. Clearly, this is not necessary for, or relevant to, our electric guitar patch. However, when you listen to someone playing an electric guitar, you hear an amplified signal through a speaker of some sort. So, while we no longer need the filter bank to model the relative position of the guitar, we now require it to model the complex response and resonances of the reproduction mechanism. Again, the patch should remain unchanged.

▶

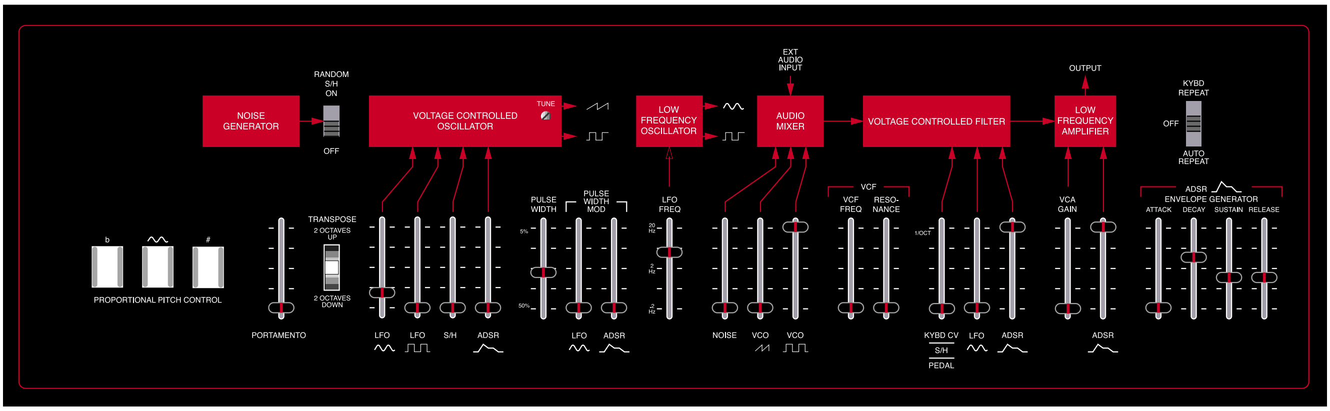


Figure 2: 'Jazz Guitar' from the ARP Axxe patch book.

The final elements in our synthesized model lie at the top of Figure 1. These control voltages and physical controllers allow us to play the sound in a realistic fashion and, in an ideal world, we will retain all of them. After all, we still want to imitate all the nuances and performance capabilities of the guitar, whether the amplifier is a hollow body or a bunch of transistors.

So where does that leave us? Surprisingly, it plonks us precisely back where we started. Although the sound of an electric guitar is very different from that of an acoustic guitar, the sound-generation mechanism is fundamentally the same.

## Patch 1 — The ARP Axxe

If I follow customary Synth Secrets practice, I should now present a second diagram that depicts the ideal electric guitar patch. At this point in our analysis, however, this would seem to be identical to Figure 1 and, since I have no wish to be ritually beheaded by *Sound On Sound*'s graphics department for gratuitous over-use of a hideously unwieldy diagram, I'm going to offer you something else. This is the 'Jazz Guitar' patch from the ARP Axxe patch book published by ARP in 1977 or thereabouts (see Figure 2 above).

As you can see, this patch is much simpler than the one depicted in Figure 1. This is inevitable; the Axxe is far too basic to do more. You can see just how simple it is by comparing the block diagram in Figure 3 (which shows all the elements of the patch) to the 100+ modules in Figure 1... Even ignoring the fact that the Axxe is monophonic, it's a huge difference.

Let's now consider some of the 'Jazz Guitar' modules, starting with the oscillator. There is an immediate surprise here ... contrary to our analysis of the past two months, the patch does not use a sawtooth waveform, but rather utilises a pulse waveform with a duty cycle of about 25 percent. This is really rather clever. The Axxe has no complex filter bank, so there is no way to remove or accentuate narrow bands of frequencies once the oscillator has generated them. Using a narrow pulse introduces some of these holes, albeit in a rather crude way. This is because a 25 percent pulse wave lacks every fourth harmonic. Sure, no guitar body has resonances as evenly spaced as the holes in a pulse wave's harmonic series, so it is — at best — a poor approximation. Furthermore, the resonances of the guitar are stationary, whereas the holes in the pulse wave's harmonic structure travel up and down the spectrum as you play up and down the keyboard. Nonetheless, the results are more satisfying than using a sawtooth, and that's what matters.

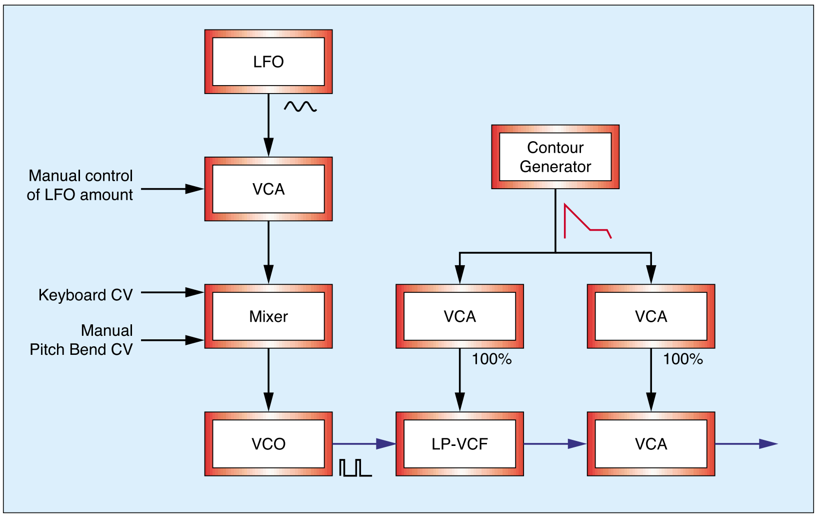


Figure 3: The block diagram of the Axxe Jazz Guitar.

Let's now look at the envelope generator. The contour generated by this is applied to both the VCF and of the VCA, thus shaping the note in the way suggested in Figure 1. However, the ADSR shape is set up rather strangely. I agree that the Attack should be as close to instantaneous as possible, and knowing the time constants of the Axxe's contours, a Decay of '6' or '7' seems reasonable. But what of the Sustain level of '4' or thereabouts? Unlike our analysis of a real guitar's contour (see Figure 4 above) the Axxe contour (shown in Figure 5, right) implies that a jazz guitar will sustain indefinitely if you ask it to do so, which — in my experience — is not the case. OK, the guitar could be heavily amplified, and the player could be

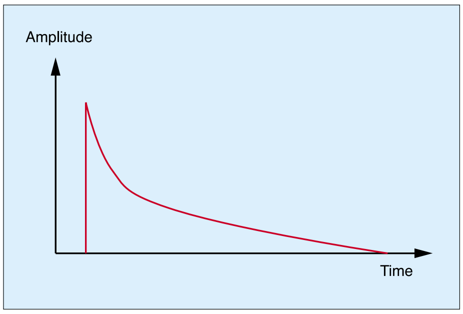


Figure 4: The amplitude and brightness contour of a guitar (from Part 28 of Synth Secrets).

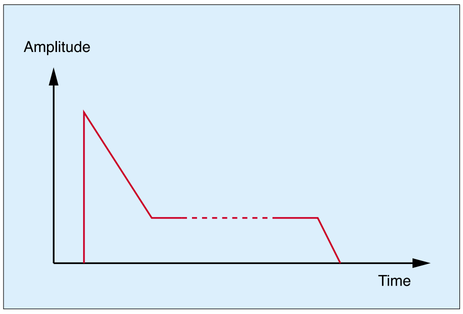


Figure 5: The contour used in the Axxe patch in Figure 2.

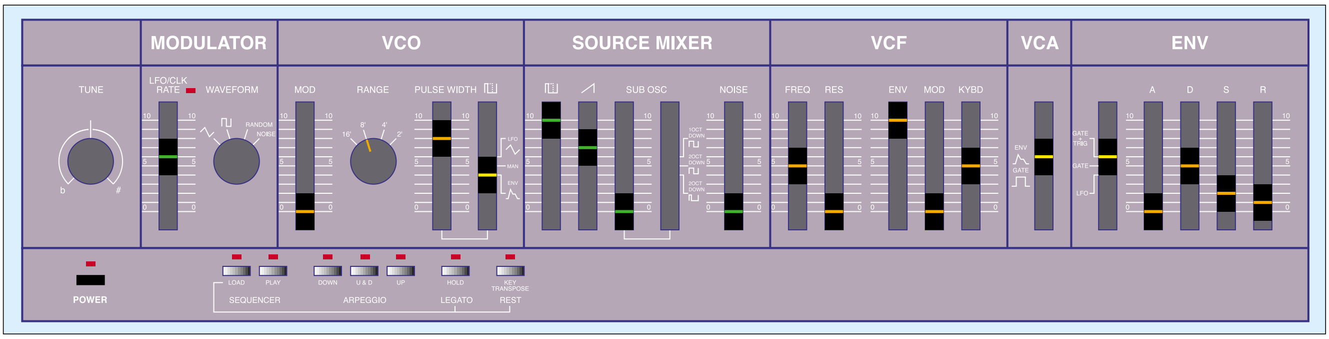


Figure 6: 'Fuzz Guitar' from the SH101 manual.

using acoustic feedback from the speaker to the strings to create indefinite sustain. But since this method of playing is more usually the province of overdriven, jaw-clenching, angst-driven, testosterone-fuelled guitar solos, I still contend that it's wrong for this sound.

Moving on, the Axxe is very limited in terms of performance capabilities, but the patch makes the best of the options available. The original version of the patch suggested that the player use the LFO slider in the oscillator section to add vibrato manually, and the pitch-bend knob to create bends and slides. The version of the Axxe shown in the diagram is significantly more expressive than this, allowing you to use the three pressure-sensitive pads (which were added to later production versions of the synth) to recreate these effects in a far more natural way.

Now, the Axxe is a monophonic synth, so there is no question of it producing strums or chords. Nevertheless, you could be forgiven for expecting satisfying results if you use it to play melodies in a fashion that is sympathetic to the realities of the electric guitar. Unfortunately, you would be wholly wrong to do so. Played dry, this patch sounds like an analogue synth. Played through a clean guitar amp, it sounds like an analogue synth played through a clean guitar amp. Played through a screaming 100W Marshall stack it sounds like... well, an analogue synth pretending to be a guitar playing through a screaming Marshall stack. The only way to get this patch to sound anything like a guitar is to distort it so much that the overriding factor is the distortion itself, not the sound produced by the synth.

## Patch 2 — The Roland SH101

Of course, it may be that the Axxe is unable to create a convincing guitar sound, so let's try another instrument. Figure 6 (above) shows the 'Fuzz Guitar' patch from the manual supplied with the Roland SH101. This shares many attributes with the patch for the ARP Axxe. For example, the pulse wave again contributes most of the sound, although this time with an initial pulse width of 80 percent. However, Roland have made the pulse width follow the envelope shape, and have added some sawtooth wave, reducing the depth of the holes in the harmonic spectrum produced by the pulse wave alone. These are nice embellishments, and they add a pleasing movement to the sound that the Axxe patch lacks. But does this patch sound like a real guitar? No, of course it doesn't.

## Patch 3 — The Minimoog

While the Axxe and the SH101 are capable of producing both sawtooth and pulse waves simultaneously, the two waveforms are merely different outputs from the same oscillator. This means that they are always in perfect phase with one another, and that the summed waveform (notwithstanding changes of pitch, or the application of effects such as vibrato) is static. So let's now look at a synthesizer that uses independent oscillators to produce these waveforms.

Figure 7 (below) is the Electric Guitar patch from Tom Rhea's *Minimoog Sound Charts* book. This mixes a pulse wave (Oscillator 2 contributing 70 percent in the mixer) with a ramp wave (Oscillator 1 contributing 50 percent in the mixer). Furthermore, this patch conforms more closely to my idea of an electric guitar, because the Sustain levels in both contour generators are zero. In addition, the decay of the filter cutoff frequency is somewhat faster than the decay of the amplifier. This attenuates the higher harmonics more quickly than the lower ones, and produces a more natural-sounding tail to each note. So, given that this is the revered Minimoog, the godfather, the source of all things wondrous and synthy... this must be the one that sounds like a real electric guitar. Right?

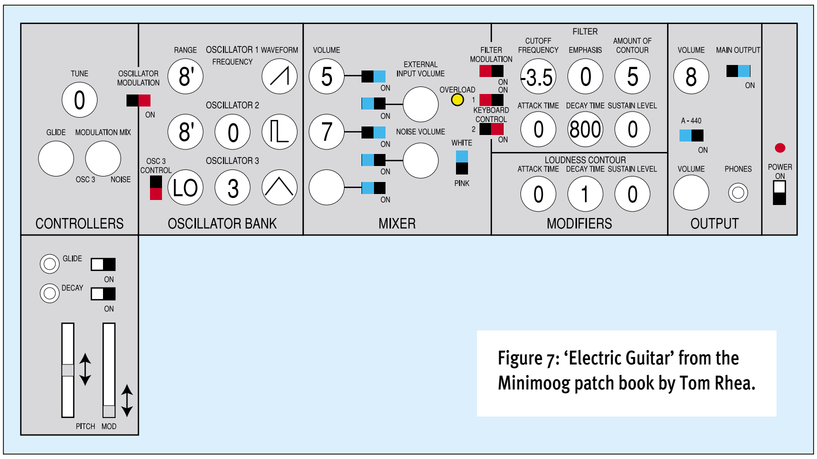


Figure 7: 'Electric Guitar' from the Minimoog patch book by Tom Rhea.

Wrong. While very pleasing to the ear (as are all three of these patches), the patch sounds not even remotely like an electric guitar. You could be lying near-catatonic on the floor after a particularly spectacular party, body suffused with harmful substances, your brain and ears filing divorce proceedings against the rest of your body... and *still* you wouldn't be fooled. Clearly, the world does not permit practical analogue, subtractive synthesis to reproduce a guitar sound. Even the sound of the electric guitar has eluded us, except in the limited and meaningless case of playing the synth through an overdriven amplifier turned up to 11 on the Richter Scale. So, to finish this three-part discussion on what has transpired to be the non-synthesis of guitar sounds, I'm going to offer you a few more clues as to why this should be the case.

## Why Don't Synths Sound Like Electric Guitars?

Let's consider something that the electric guitar has, but which an acoustic guitar does not — pickups.

Figure 8 (right) shows a typical guitar pickup. This comprises a magnetised core with a coil of wire wrapped around it, all placed underneath but close to the metal wire that we commonly refer to as an electric guitar 'string'. Some basic physics (which we need not discuss) tells us that an electrical conductor vibrating in a magnetic field excites an electrical current, and it is this signal that is amplified and passed to the speaker for us to hear.

At this point, let's consider two things that seem unrelated but which, when taken together, prove to be vitally important. The first is: the distance over which the pickup can detect the string's motion is very short, so it is only sensitive to the small part of the string that lies immediately above it. The second is: when the string is stationary along the length detected by the pickup, no signal is generated.

Now consider the case where the pickup lies, say, one third of the distance from the bridge to the nut, as shown in Figure 8. If you consider the wave motion, you will realise that the first harmonic (the fundamental) and the second harmonic each exhibit some motion immediately above the pickup. However, the third harmonic has a node at this position (ie. it is permanently stationary) and, therefore, this frequency — which is present in the acoustic signal — will be missing from the amplified signal (if this isn't clear, refer back to Part 28 of this series, in *SOS* August 2001, or at www.sound-on-sound.com/sos/aug01/articles/synthsecrets28.asp, which explains the importance of nodes in this context). Likewise, the sixth, ninth and other harmonics in this series will be missing.

Now consider Figure 9 (above), which shows what happens when you play a note one third of the way along the *neck*. In this case, it's the *second* harmonic that is stationary above the pickup, along with the fourth, sixth, eighth... and so on. In other words, the pickup position acts as a comb filter whose harmonic spacing is dependent upon the pitch of the note you are playing.

Clearly, the relative amplitudes of the harmonics are greater when there is an anti-node (a position of maximum movement) immediately above the pickup, and less if a node lies above or close to the pickup. This means that the harmonic structure of the amplified signal is determined by the position of the pickup, as well as by the acoustic vibration of the string.

As an example of this, Figure 10 (below) shows the artificially simple case in which a sawtooth wave is 'played' through a pickup. As expected, the result is much like that of passing the wave through a comb filter.

Now let's view the effect of the pickup on the spectrum of an electric guitar (see Figure 11 below). The resulting harmonic structure is singularly unlike anything produced by our affordable subtractive synthesizers, so it's not surprising that our Axxe, SH101 and Minimoog patches lack realism.

By the way, you may have noticed that the low-frequency region of the spectrum in Figure 11 is much flatter than that of the common analogue waveforms (most of these conform to the 1/n amplitude relationship we've discussed before). This is because, on the electric guitar, the output of any harmonic is proportional to the velocity of its motion at the point on the string that lies immediately above the pickup. If you can picture the position of a bridge pickup and consider the displacements and velocities of the first few harmonics, you'll realise that the amplitudes produced by each are similar (don't worry if you can't envisage this... trust me). This means that until you approach the first harmonic with a node positioned above the pickup, the spectrum can be surprisingly flat.

Finally, I want to return to something that I touched very briefly upon in Part 28, where I stated that "the strings are not perfect harmonic oscillators. This is because they do not form perfect angles at the nut or the bridge. The finite cross-section of the string ensures that it curves at these points, thus making its active length slightly shorter than the idealised view would suggest. The degree to which this happens depends upon the wavelength and amplitude of the vibration, so the string appears shorter at high frequencies and high amplitudes than it does at low frequencies and low amplitudes. This sharpens higher, louder harmonics, further complicating any analysis of the string's harmonic spectrum, as well as the guitar body's response to it."

Figure 12 shows (in grossly exaggerated form)

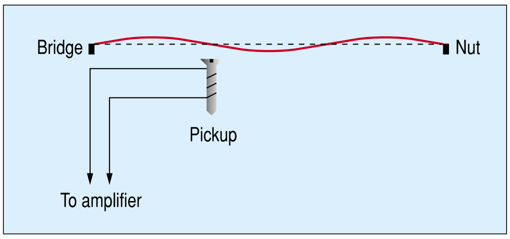


Figure 8: A simplified view of one guitar pickup and string.

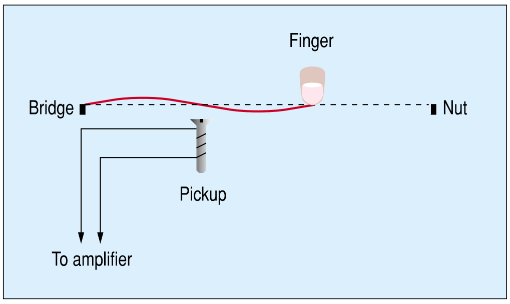


Figure 9: Changing the pitch of the note.

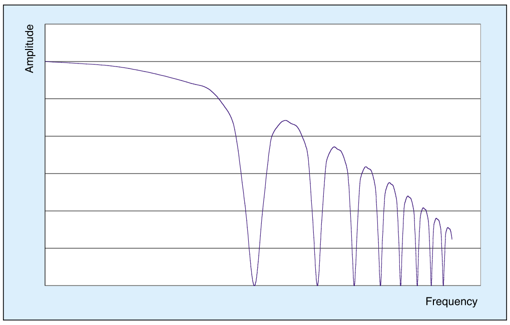


Figure 10: The effect of a guitar pickup on a sawtooth wave's harmonic spectrum.

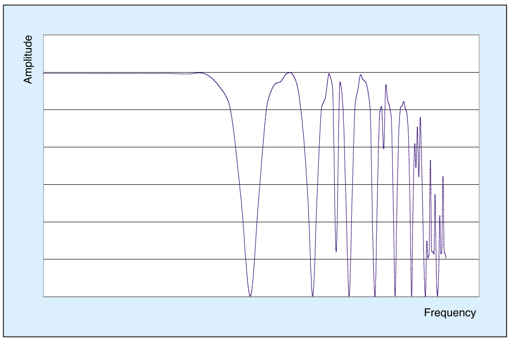


Figure 11: The effect of the pickup on a guitar's harmonic spectrum.

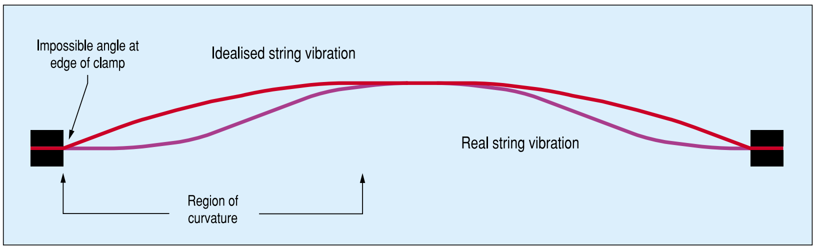


**Figure 12: A hugely exaggerated representation of why strings are not perfect oscillators.**

how this would be true for a string clamped at both ends. However, just to complicate matters still further, you should be able to see that it is only true on a guitar when the string bends 'down' with respect to the bridge and/or the nut. When it bends 'up' it lifts slightly at the edge of the nut and bridge, thus extending its effective length, and flattening the pitch for half the cycle. As for sideways and other angles… don't even ask. You don't want to know.

Anyway, consider the start of a note of any given pitch. You have just plucked the string, so its effective length is somewhat shortened by the initial high amplitude, and the overtones are *enharmonic*, meaning that they do not conform to the pure 1, 2, 3, 4… relationship of synthesizer oscillators. As the loudness of the sound decays, the effective length of the string becomes longer, and the pitch decreases.

As the amplitude decays further, the overtones become more nearly harmonic. However, while this is happening, the positions of the nodes and anti-nodes are sliding down the string, changing their positions with respect to the pickups, and thus modifying the harmonic mix of the electrical audio signal as they do so (by the way, it is this effect that the SH101 imitates by modulating the pulse width using the envelope generator. I wonder whether Roland's patch designers knew).

## Conclusions

As you can see, the electric guitar is every bit as complex and unfathomable as the acoustic guitar. Indeed, it is even *more* complex and unfathomable because, although the body resonances and their interactions with the strings are less significant, there are whole new families of effects to consider. All this… and we haven't touched upon multiple pickups, in-phase and out-of-phase pickup configurations, the phase shifts introduced by tone controls, and the capacitances of the cables (which can be very significant for such small signals).

At this point, I would like to draw Figure 13 for you to show some of this, adding the amplitude- and pitch- dependent comb filters and other effects. However, it wouldn't fit on an A4 page. So I think it's time to admit defeat and accept that, as I stated two months ago, we're never going to effectively imitate guitars using analogue synthesizers. SOS

Photos: Richard Ecclestone, Piers Allardyce, Nigel Humberstone

# Synth Secrets

**Synth Secrets turns its attention to the synthesis of percussion instruments, beginning with pitched drums.**

Gordon Reid

## Synthesizing Percussion

Over the past seven months, I've been applying the principles of synthesis to recreate the sounds of two, very important classes of musical instruments: those whose primary oscillators are blown pipes and plucked strings. Having discovered that it's possible to synthesize the first of these classes quite well, and the second class not at all, it's now time to turn elsewhere. This month, I'll begin looking into the sounds and synthesis of percussion instruments.

### What Are Percussion Instruments?

It's tempting to think in rather narrow terms when discussing percussion instruments. However, given that these are the oldest musical instruments developed by mankind, it shouldn't surprise you to discover that there are more instruments in the percussion family than there are in any other.

I'll start with a definition: a percussion instrument is one that produces its sound when the player strikes some part of it. Mind you, such definitions are almost inevitably incomplete. For example, when the drummer in a jazz trio plays the snare drum by sweeping brushes on its upper skin, it is still a percussion instrument. Similarly, when a violinist creates a musical effect by striking the strings with the bow, the violin is not a percussion instrument, despite being played percussively.

Ignoring such annoying complications, I will follow the academic practice established in some texts and define four primary percussion groups. These are the *membranophones* (more commonly called drums), the *idiophones* (instruments such as glockenspiels and xylophones, plus metallic percussion such as cymbals, high-hats and gongs), the *aerophones* (whistles), and the *chordophones* (primarily the piano family). You might think it odd that the piano is considered a percussion instrument, but then how else could you reasonably classify something that uses hammers to produce its sounds?

Having divided percussion instruments into these primary groups, it's possible to further sub-divide the membranophones and idiophones into four sub-groups: those that produce pitched sounds, and those that do not. For example, kettle drums (or 'timpani') are *pitched* membranophones, whereas bass drums, snare drums, and toms are not. Likewise, bells, glockenspiels, xylophones and even gongs are pitched idiophones, whereas cymbals and high-hats are not. It's not necessary to sub-divide the aerophones and chordophones in this way, because these use columns of air and strings as their respective primary resonators, so it's safe to assume that they generate pitched sounds.

But you can also sub-divide the membranophones in a different way; by considering their physical construction. Some — such as timpani — have a single membrane and an air cavity enclosed by a rigid body. Others — such as toms and congas — have a single membrane with air at atmospheric pressure on both sides. Yet a third sub-class — bass drums, snares and so on — have enclosed air trapped between *two* membranes.

Likewise, you can further sub-divide the idiophones into shaken instruments (rattles), scraped instruments (rasps), concussion instruments (claves) and struck instruments (such as the aforementioned glockenspiels and xylophones). I strongly suspect that there are other sub-sub-groups, but I'm sure you get the general idea.

So there are numerous groups of percussion to consider — certainly far too many for a single instalment of Synth Secrets. Indeed, there are whole academic text books devoted to the subject, and, while these make a brave stab at covering the physics of each group, they do not attempt to describe the synthesis of the instruments' sounds. Therefore, rather than try to achieve the impossible — ie. a complete description of all percussive instruments — I'm going to take a single sub-class, and attempt to synthesize this. The sub-class is the pitched membranophone, and the specific instrument is the kettle drum.

### Pitched Drums

Before proceeding, I'm going to jump back a couple of years to Part 2 of this series, and recap some of the basic acoustics of the circular membrane (see *Sound On Sound* June '99, or surf to www.sound-on-sound.com/sos/jun99/articles/synthsecrets.htm). Incidentally, it's important to remember that every time I say 'circular membrane', it's shorthand for 'a circular membrane fixed at its edge'. This is because the physics of the unsupported membrane is quite different from that of the fixed one.

If you can remember that far back, you'll recall that the difference between a harmonic oscillator (such as a stretched string) and a circular membrane is one of dimensions: the string has just one, whereas the membrane has two. The consequence of this is that, whereas the overtones of the string occur at integer multiples of the fundamental frequency 'f' and therefore sound pitched or musical to our ears, the overtones of the membrane occur at 'enharmonic' frequencies that hold no obvious relationship to the fundamental, and therefore rarely sound musical to us. Of course, while it may not be obvious, the ratios of the frequencies are far from haphazard; they are determined by those hideously complex equations, the Bessel Functions, as described in Part 2 of this series.

Let's now look at some of the membrane's simpler modes of vibration, using the terminology from Part 28 of this series (see *SOS* August '01, or head to www.sound-on-sound.com/sos/aug01/articles/synthsecrets28.asp). The simplest is the 0,1 mode. This has no radial modes of vibration (ie. ones whose nodes, or points of zero displacement, bisect the centre of the membrane), and just one circular mode, with its node at the fixed edge (see Figure 1, right).

The next two modes lacking radial components are, as you might expect, called the 0,2 and 0,3 modes, and these have frequencies of 2.30f and 3.60f respectively (see Figures 2 and 3 below).

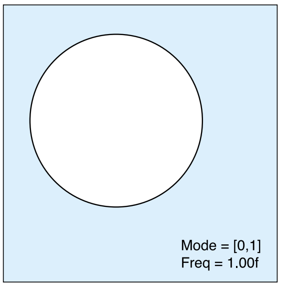
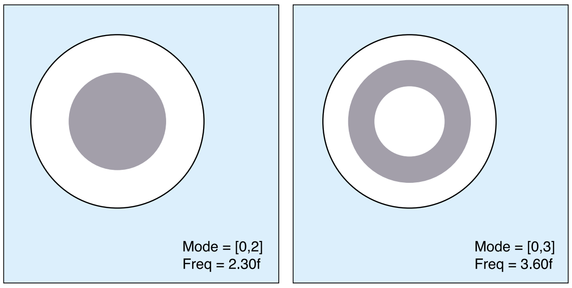
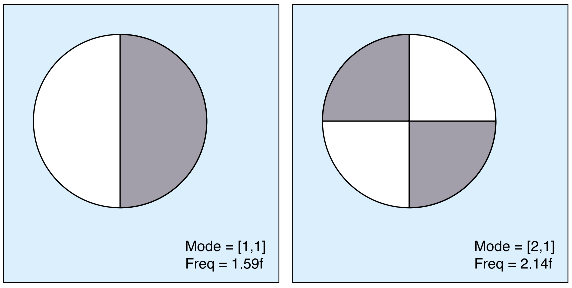


Figure 1: The 0,1 mode — the fundamental — with frequency ratio 1.00f. To understand this and the other diagrams in this article, imagine that you are looking down on a drum skin and that when the white parts are vibrating 'up' at any given moment, the grey parts are vibrating 'down', and *vice versa*.



Figures 2 and 3: The 0,2 and 0,3 modes, with frequencies of 2.30f and 3.60f respectively.



Figures 4 and 5: The 1,1, and 2,1 modes, with frequencies of 1.59f and 2.14f respectively.

So what about the radial modes? The 1,1 and 2,1 modes are very simple, as you can see from Figures 4 and 5 (above), but things start to look decidedly more complex when you mix circular and radial modes, such as 2,2 (see Figure 6, on the next page).

If I now take the frequencies of a selection of low-order modes and chart their relative relationships (see Table 1,

▶

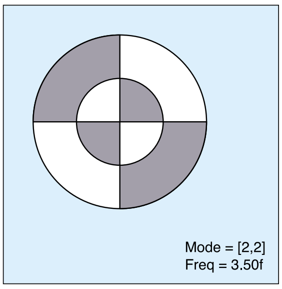


Figure 6: The 2,2 mode, with frequency 3.50f.

▶ below), you can see that the result is quite unlike that of a string or pipe. As you already know, the circular membrane is *not* a harmonic oscillator.

| MODE | FREQUENCY RELATIVE TO THE FUNDAMENTAL |
|---|---|
| 0,1 | 1.00 |
| 1,1 | 1.59 |
| 2,1 | 2.14 |
| 0,2 | 2.30 |
| 3,1 | 2.65 |
| 1,2 | 2.92 |
| 4,1 | 3.16 |
| 2,2 | 3.50 |
| 0,3 | 3.60 |
| 5,1 | 3.65 |
| 3,2 | 4.06 |
| 6,1 | 4.15 |

Table 1: The relative frequencies of 12 low-order modes of a circular membrane.

Since numerous modes will be excited every time you strike the membrane, it's likely that all the modes in Figure 7 below (and many others not shown), will be present when you hit the skin of a membranophone. Moreover, since the relationships of the frequencies are enharmonic, it's equally likely that, unless one mode dominates to the exclusion of all others, you will hear an atonal sound with no recognisable pitch. Hang on... didn't I say that the idea was to synthesize *pitched* membranophones? What's going on?

To answer this, it's necessary to investigate the nature of the instrument itself. In Part 2 of this series, I considered the physics of the membrane itself. But a kettle drum introduces many other factors...

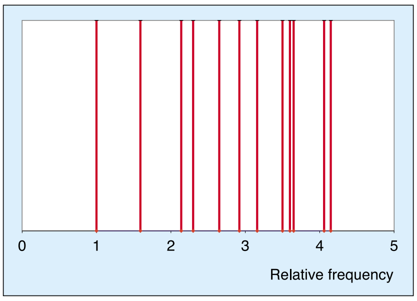


Figure 7: The relative frequencies of the 12 low-order modes, depicted graphically.

There is atmosphere on one side of the membrane, there are the lumps of wood and metal called the shell of the drum, and there's an enclosed volume of air within the drum itself. So, stepping far beyond our previous discussions of circular membranes, it's time to look at the drum as a whole.

## The Kettle Drum — Initial Spectrum

Although it has long been known that drums are essentially enharmonic, the tonal nature of timpani was understood more than two centuries ago. Researchers recognised that the 1,1 mode was producing the principal frequency, with two strong overtones at ratios of 3:2 (a perfect fifth), and 2:1 (an octave) above the principal. This is an amazing result, not least because the principal is not the fundamental, which lies at approximately 63 percent of the principal frequency. But the most important question this raises is; how have the enharmonic modes suddenly become harmonic?

Let's look at the frequencies of these overtones in terms of the fundamental frequency, 'f'. If the principal is the 1,1 mode then, from Table 1, you can see that this lies at a frequency of 1.59f. Consequently, the observed overtone that lies a fifth above the principal must be at approximately 2.39f, and the octave overtone must lie at 3.18f (ie. double the principal frequency). The ideal membrane has natural modes that lie close to these frequencies, but it is one thing to be *close to*, and quite another to be *at*. Furthermore, you might well ask what has happened to all the other modes. To understand the mechanisms involved, you must consider five primary factors:

- Firstly, the membrane no longer vibrates in isolation: one side is enclosed by air;
- Secondly, the other side of the membrane is enclosed by the air within the shell;
- Thirdly, the kettle is a resonant volume, so the air within it will itself have resonances that interact with the vibrations of the membrane;
- Fourthly, the membrane acts like the string in Figure 12 of last month's Synth Secrets... it has a finite stiffness at its edges. This means that it appears slightly smaller at higher frequencies (and higher amplitudes) than at lower ones, thus sharpening the partials produced by the higher modes;
- Fifthly, stretched membranes don't like to wrinkle, so they are resistant to the vibrational modes that attempt to make them do so (the 0,1 mode is one of these, as you can probably imagine if you mentally increase the magnitude of the displacement until the membrane looks hemispherical; consider the wrinkles you would create at the edge of a circular piece of paper if you wrapped it over the top of a globe, and you'll see what I mean).

If you think that several of these factors look familiar, you're right. The kettle drum shares many attributes with the acoustic guitars I've been discussing recently. Admittedly, there are significant differences: in particular, the strings of the guitar are harmonic, and its body has a sound hole. Nonetheless, the similarities are striking: in both cases you excite a primary oscillator which interacts with a complex set of air resonances, and the resulting vibrations radiate outwards as the sound you hear.

| MODE | FREQUENCY OF IDEAL MEMBRANE RELATIVE TO FUNDAMENTAL | FREQUENCY RELATIVE TO PRINCIPAL | FREQUENCY RELATIVE TO PRINCIPAL OF MEMBRANE IN AIR | FREQUENCY SHIFT RESULTING FROM SUSPENSION IN AIR (%) |
|---|---|---|---|---|
| 1,1 | 1.59 | 1.00 | 1.00 | 0 |
| 2,1 | 2.14 | 1.34 | 1.47 | +10 |
| 3,1 | 2.65 | 1.66 | 1.91 | +15 |
| 4,1 | 3.16 | 1.98 | 2.36 | +19 |

Table 2: A selection of the radial modes of a real membrane suspended in open air, and their relative frequencies.

A kettle drum also has a hole that stops the membrane bulging or bowing as the weather — and therefore the atmospheric pressure — changes. However this is small, is located at the bottom, and has little effect on the sound.

At this point, let's take our hypothetical ideal membrane, suspend it in free air, strike it, and measure the frequencies of the resulting modes. In practice, the action of moving the air on each side shifts the modes considerably, raising the radial modes much closer to harmonic relationships (see Table 2, above). They're still too flat to fool the ear into thinking that it's listening to a true harmonic oscillator, but the sound nevertheless conveys a strong sense of tonality without ever quite ▶

sounding like a pure tone.

If you now take this membrane and further enclose one side by attaching it to the rim of a kettle drum, you force it to interact with the modes of vibration of the air within the body of the drum. The physics of this is complex, but it should be relatively simple to see that, if a mode in the trapped air has the same 'shape' at its boundary with the membrane as one of the membrane's own modes, they will interact strongly. For example, the air mode in the kettle drum in Figure 8 (shown below) will interact with the 1,1 mode of the membrane from Figure 4. Indeed, the membrane's 1,1 mode is instrumental in exciting the air mode shown, and if their frequencies are similar, they will interact even more strongly, reinforcing one another until their energy is dissipated.

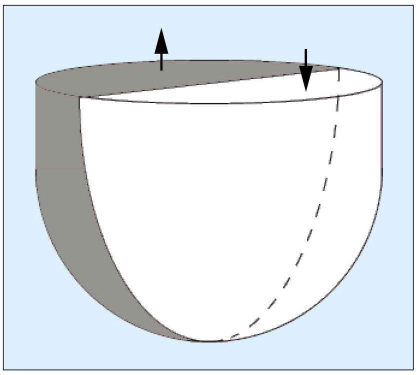


Figure 8: The enclosed air mode that interacts with the membrane's 1,1 mode.

Experiments show that the air modes that couple to the n,1 membrane modes are always of higher frequency than the membrane modes themselves. The air modes therefore 'drag' the membrane modes to even higher frequencies than those shown in Table 2, sharpening them further, and bringing them very close to the harmonic ideal (see Table 3, below).

Now, you might wonder why I have only placed radial modes into Tables 2 and 3. There's a very good reason for this: these are the only ones that become harmonic; the circular modes do not. Knowing this, a skilled timpanist will ensure that the sound of his instrument is as musical as possible by striking the membrane almost precisely a quarter the way from the edge to the centre. In doing so, the timpanist suppresses the circular modes, ensuring that the quasi-harmonic radial modes dominate the sound (see Figure 9, below). On the other hand, striking a kettle drum in the centre of its membrane depresses (by and large) the radial modes in favour of the enharmonic (and, as we shall see in a moment, short-lived) circular modes. So, if you want to produce a dull, toneless, and musically uninteresting thump, hitting the timp dead centre is the way to do it.

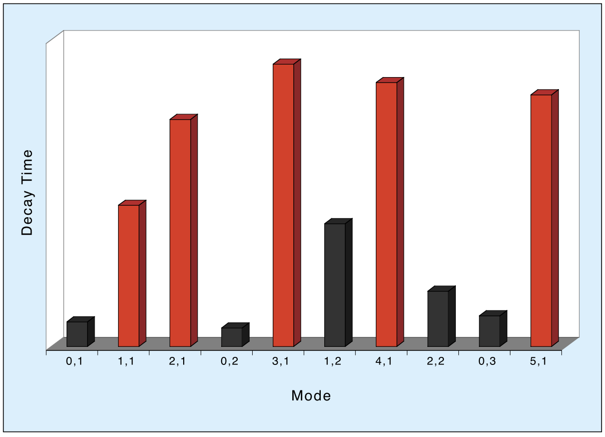


Figure 10: How the decay times of the kettle drum's modes differ.

## The Kettle Drum — The Shape Of The Sound

As you can see from the differences in Tables 2 and 3, the body of the kettle drum acts as a fine-tuning agent that forces the overtones of the membrane into an almost perfect harmonic series. However, it does far more than this and (like an acoustic guitar enclosure) has a strong effect on the duration and decay of the sound.

Consider this: a rigid body like the shell of a drum increases the ability of the attached membrane to radiate sound into the atmosphere. This is a good thing for an instrument, because it means that more of us, spread over a larger area, can hear it. This isn't so important in today's era of clip-on microphones and 10,000W PA systems, but it was vital for the orchestras of the 18th and 19th centuries.

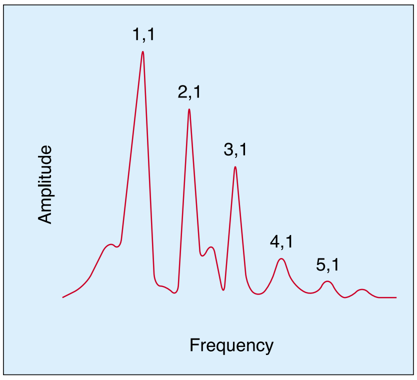


Figure 9: A representation of the initial spectrum of a correctly played kettle drum.

| MODE | FREQUENCY OF IDEAL MEMBRANE RELATIVE TO FUNDAMENTAL | FREQUENCY RELATIVE TO PRINCIPAL | FREQUENCY RELATIVE TO PRINCIPAL OF TIMPANI | FREQUENCY SHIFT RESULTING FROM ATTACHING TIMPANI (%) |
|---|---|---|---|---|
| 1,1 | 1.59 | 1.00 | 1.00 | 0 |
| 2,1 | 2.14 | 1.35 | 1.50 | +11 |
| 3,1 | 2.65 | 1.67 | 1.98 | +19 |
| 4,1 | 3.16 | 1.98 | 2.44 | +23 |

Table 3: A selection of the radial modes of a real kettle drum, and their relative frequencies.

However, if an instrument radiates its energy more quickly, the sound will — for obvious reasons — decay more rapidly. What's more, the increase in radiative capability is not consistent across the spectrum and, while most modes will decay more rapidly than they would otherwise do, some will decay *far* more rapidly. This leads to a change in the harmonic structure of the sound as the note progresses.

Figure 10 (shown above) represents the decay times for a number of low-order modes in a kettle drum. As you can see, the desirable radial modes have long decay times compared with the undesired, enharmonic, circular modes. This means that, when you strike a kettle drum, there is a short period of enharmonicity — that is, a burst of atonal sound that is much like noise — followed by a longer period of the tonal sound associated with the instrument. This information may be represented in a diagram called a spectrogram, as shown in Figure 11 (above right).

Since it's possible that you haven't encountered a spectrogram before, I'll explain what Figure 11 represents. The vertical axis is the frequency axis, and the horizontal axis depicts time. Therefore, at any position along the time axis, you can look up the chart to see what frequencies are present, and in what proportions. Clearly, at the start of the 'note', many frequencies are present. However, these decay rapidly, allowing the quasi-harmonic modes to emerge as the dominant sound before they too decay.

This is all relatively straightforward, but look to the far right of the spectrogram. Just before the note decays to silence, the principal 1,1 mode vanishes, leaving the 2,1 and 3,1 modes to linger. This means that the perceived pitch of the note may leap a fifth or even an octave as it decays. Just to complicate matters still further, there are psychoacoustic effects that, under certain conditions, make the brain 'reinsert' the missing principal mode. Unfortunately, or maybe fortunately, there's no space to describe those effects here.

## Other Tuned Membranophones

It should come as no surprise to learn that timpani are not the only pitched membranophones, although others are relatively rare in European music. However, tuned percussion has long been of greater importance in cultures further east than it has in European musical traditions. There are many pitched Indian percussion instruments, and these are enjoying a surge of interest in electronic music. You only have to inspect the waveforms stored in the ROMs of every modern samploid synth and workstation to see how omnipresent the sounds of tablas and similar Indian instruments have become. However, while these share many properties with timpani, they are very different instruments... they're smaller, lighter, and played with the hands rather than beater sticks.

The small size of these instruments has a very important effect: it reduces the effect of the air loading. This means that, if I measured the membrane's modes and inserted the results in another table, they would more closely represent the Bessel functions of the theoretical ideal than does the large membrane shown in Table 2, and so produce less musical, tonally enharmonic overtones. Consequently, manufacturers will load the membrane with materials such as gum or even sticky rice laced with minerals. Once applied, these stiffen the drumhead and add weight, increasing the frequencies of the partials until they more closely resemble harmonics. If a single application is insufficient, it's likely that multiple layers will be added until the desired tonality is achieved.

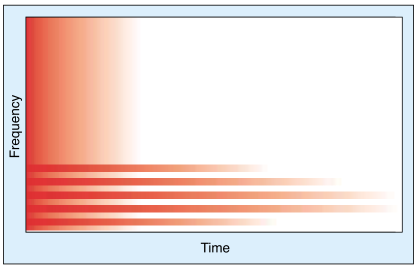


Figure 11: A spectrogram of the sound of timpani.

## Pitched Membranophones In Performance

At this point, you might feel that I've described everything necessary to synthesize the sound of the pitched membranophone, but sadly, this is not the case. Just as in the discussion of the acoustic guitar, the relative amplitudes of the modes that you hear will depend upon the position from which you're listening to the drum. This is because each of the modes possesses a different radiation pattern. So, while in one position you may hear lots of 2,1 and very little 3,1 mode, in another position the situation could be reversed. This means that, at any given moment, no two listeners hear precisely the same sound. This is particularly apparent in the open air, whereas a concert hall will reduce the differences in perceived sound by bouncing the partials around its walls and other surfaces. Unfortunately, as discussed in Synth Secrets 22 (see *SOS* February '01, or www.sound-on-sound.com/sos/feb01/articles/synthsecrets.asp) this leads to other problems, as the room imposes its own modes upon the sound generated by the instrument.

Finally, before you can synthesize the sound of pitched membranophones, it's important to consider one more aspect that defines their sounds: the performance. Some drums, when hit harder, initially produce a higher pitch than they do when hit softly. This is largely a consequence of the fourth factor described earlier: ie. that the membrane appears slightly smaller at higher amplitudes than it does at lower amplitudes. However, this is far less relevant to a kettle drum than it is to other drums. This is because the air modes within the cavity act as stabilisers, forcing the oscillations to the equilibrium frequencies determined by their interactions with the membrane modes. This means that, when considering the striking force, only the amplitude of the resulting sound is important, not the frequency or harmonic spectrum.

Nonetheless, most pitched membranophones can be played over a small range of frequencies. On Eastern instruments, the player controls this by exerting pressure on the membrane with the palm of the playing hand. On timpani, the tuning mechanism is pedal-operated, and offers a range of approximately half an octave. So when you come to synthesize the pitched membranophones, you need to incorporate a way to change the pitch smoothly over a small range. However, given that I've now run out of space, that will have to wait until next month. SOS

# Synth Secrets

**Building on the theory of what makes up the sound of timpani, as explained last month, this month's Synth Secrets reveals how to synthesize realistic kettle drums using a Korg MS20 and a Yamaha DX7.**

## Practical Percussion Synthesis

Gordon Reid

Last month, I spent the whole of Synth Secrets introducing the various families of percussion instruments, and conducting a detailed analysis of one small sub-class of these: the pitched membranophones. This month, having primed you with all the theory of kettle drums (or timpani), I'll investigate a number of ways to translate what you have learned into practical synthesis.

### More About Spectrograms

I'll start with last month's Figure 11, reproduced here as Figure 1. This is a simplified and idealised spectrogram of the sound of a single kettle drum being struck once.

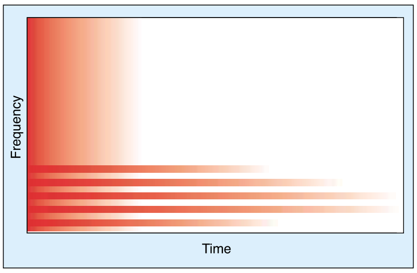


Figure 1: A simplified spectrogram of the sound of a kettle drum.

Spectrograms are very useful because, although drawn in two dimensions, they represent all three dimensions of sound: time, frequency, and loudness. OK, it's easy to see the time and frequency axes, but where is the loudness information? Well, in my diagram it's represented by the brightness of the red at any given point on the chart. I'll give you an example: the point marked '1' in Figure 2 below represents a sound of frequency 150Hz, measured 0.1 seconds after the start of the spectrogram, with an intensity of around 95 percent of maximum amplitude. Likewise, the point marked '2' represents a component of frequency 750Hz, 0.3 seconds after the start of the spectrogram, with an intensity of about 35 percent. Of course, it's highly unlikely that all the components of the sound would start at the same amplitude, but I'll address that issue later.

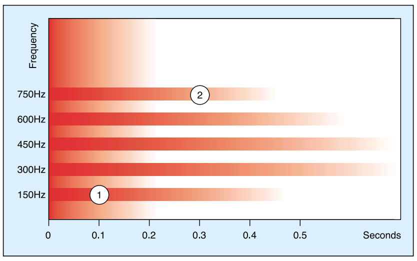


Figure 2: Understanding the spectrogram.

Now that you understand what Figure 1 is telling you, you should be able to 'see' how a single strike of the kettle drum sounds. You should also be able to hypothesise that the sound produced by striking the drum three times (thus re-energising the sound on each occasion) will look like Figure 3 below.

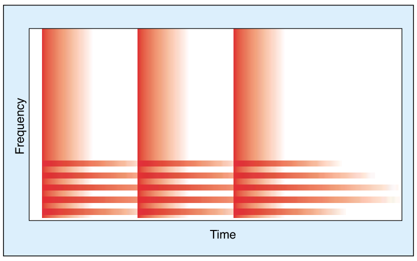


Figure 3: Striking the same kettle drum three times.

### Modelling Timps — Pitched Modes

These spectrograms give a strong clue about how to model the sound of timpani. As you can see, Figure 3 depicts five strong partials in the sound, plus a wash of frequencies that decay more quickly than those partials.

I don't know about you, but Figure 3 screams 'additive synthesis' to me. I first introduced this in part 14 of this series (see *SOS* June 2000 or www.sound-on-sound.com/sos/jun00/articles/synthsec.htm), at the end of which I drew the block diagram shown in Figure 4 on the next page. This architecture may look complex, but it's not. Starting from the left, there's a single trigger which, in the real world, will be provided by a keyboard. This trigger initiates the action of six contour generators. Four of these control the gain of Amplifiers 1 to 4, which pass the outputs from Oscillators 1 to 4. The other two control the cutoff frequency of a filter and the gain of an amplifier that, in turn, determine the tone and loudness of the contribution of a noise generator.

Let's now choose the frequencies and waveforms for the oscillators. This is simple: you know from last month that the relative frequencies of the first four radial modes are as shown in Table 1 below, so, if you know the frequency of the principal 1,1 mode, you can calculate the frequencies of the others.

| MODE | FREQUENCY RELATIVE TO PRINCIPAL OF TIMPANI |
|---|---|
| 1,1 | 1.00 |
| 2,1 | 1.50 |
| 3,1 | 1.98 |
| 4,1 | 2.44 |

Table 1: The most important partials of a real kettle drum.

Measurements of real timpani show that the principal frequency of a large kettle drum is higher than you might expect. In these days of super-deep analogue thumps and almost subsonic bass, a kettle drum has a (not particularly deep) principal frequency of around 150Hz. This is all you need to work out the frequencies of the first four modes (they're shown in Table 2 on the next page). Furthermore, because I'm talking about individual modes of vibration, the waveforms are sine waves.

Now let's consider contour generators 1 to 4. Looking at the spectrogram, it's clear that each should generate a simple

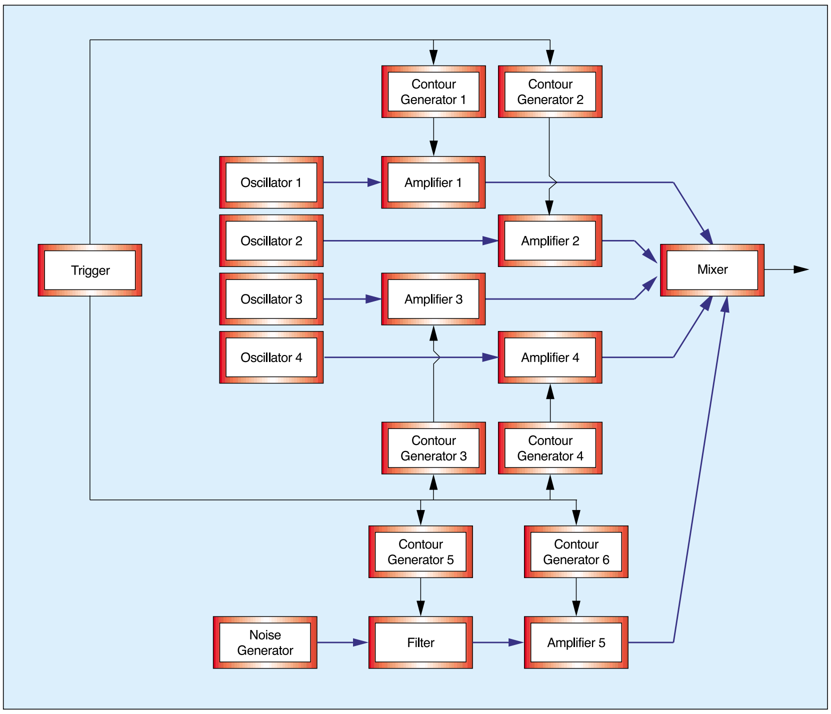


Figure 4: An additive synthesizer, as introduced in part 14 of this series.

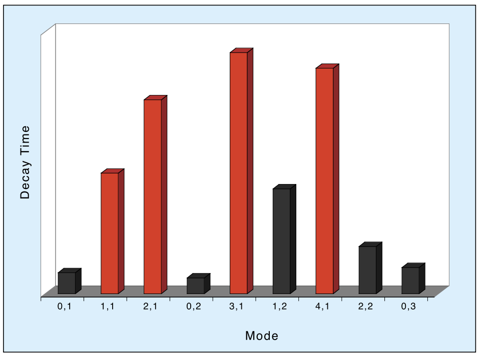


Figure 5: The decay times of the first nine kettle drum partials.

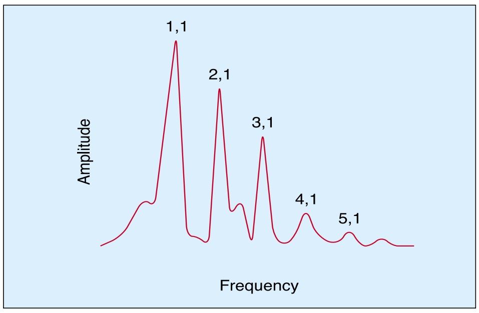


Figure 6: An approximation to the relative amplitudes of the major kettle drum partials.

Attack/Decay/Release contour with Attack = 0, and equal Decay/Release rates (ensuring that each contour completes its decay whether you hold the key or release it before each note is concluded). However, it's also clear that the four contours should have different values for the Decay/Release.

| MODE | FREQUENCY RELATIVE TO PRINCIPAL OF TIMPANI | FREQUENCY IN HZ |
|---|---|---|
| 1,1 | 1.00 | 150 |
| 2,1 | 1.50 | 225 |
| 3,1 | 1.98 | 297 |
| 4,1 | 2.44 | 366 |

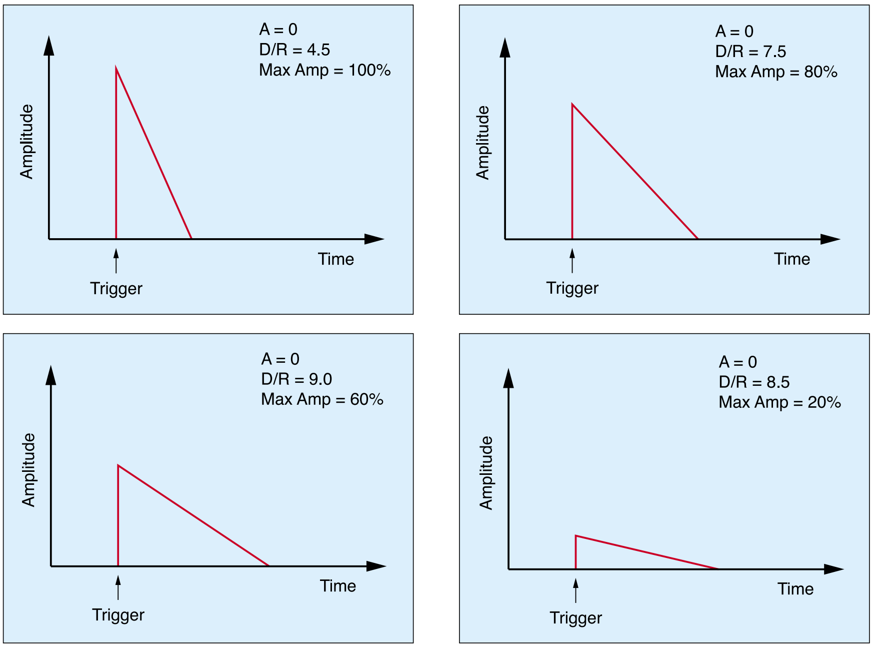
Table 2: The frequencies of the first four kettle drum partials.

Figure 5 above shows idealised decay times for the first nine modes, of which the ones in red are the quasi-harmonic partials. Now, I happen to know (because I drew the diagram) that the relative decay/release times for the first four quasi-harmonic partials are approximately 45 percent, 73 percent, 91 percent, and 84 percent respectively. You might think that this is all the information you need before defining the precise contour for each mode, but before you can determine these, you need to know the relative amplitudes of each. For that, I refer you back to another of last month's diagrams, reproduced here as Figure 6 above.

This diagram suggests that amplitude ratios of approximately 5:4:3:1 would be appropriate for the first four modes. Knowing this, you can set your four contour generators as shown in Figures 7(a) to (d) on the next page.

## Modelling Timps — Enharmonic Modes

Now that you've defined the components supplying the pitched part of the sound, you need to define the burst of noise at its start.



Figures 7(a) to 7(d): The contours of the four most important modes.

Here's an admission: last month I was lazy. Instead of drawing myriad little lines of low amplitude to represent all the enharmonic modes generated when you hit a kettle drum, I used my graphics package to draw something similar: a burst of white noise. But that's not what you get when you strike a membrane. You get myriad enharmonic partials. So what can you do to emulate this correctly? The solution is surprisingly simple. You can use a ring modulator (a form of amplitude modulation, 'AM') or frequency modulation, 'FM'.

Let's consider ring modulation, simply because this is the means most likely to lie at your disposal. If you cast your mind back to part 11 of this series (see *SOS* March 2000 or www.sound-on-sound.com/sos/mar00/articles/synthsecrets.htm), you'll remember that this is a special case of Amplitude Modulation. As such, it uses two signals (which I'll call '1' and '2'), and produces side-bands according to the complex equation shown below, where $w_1$ is the frequency of signal 1, $w_2$ is the frequency of signal 2, $a_1$ is the amplitude of signal 1, and $a_2$ is the amplitude of signal 2.

$$A_1 = a_1\cos(w_1t) + 1/2[a_2\cos(w_1+w_2)t] + 1/2[a_2\cos(w_1-w_2)t]$$

If the modulator and carrier signals are merely sine waves (which, remember, have no harmonics), the result is not very useful; it's simply the carrier, plus two signals of frequency (w1 + w2) and (w1 - w2). However, if you set both the modulator and the carrier to be harmonically rich sawtooth waves, the result is two complete harmonic series that interact according to that complex equation.

Let me demonstrate the consequences of this by calculating a matrix that shows the 50 partials generated by the interaction of just the first five harmonics of each waveform (see Table 3 below).

If I show Table 3 as a chart of frequencies (as in Figure 8 below), you can see that the results are far from harmonic, with many clumps and clusters of partials. All this, and I only calculated the outputs generated by the first five harmonics in each signal. Given that, within the audible 20kHz audio spectrum, there are 230 harmonics of an 87Hz signal, and 200 harmonics of a 100Hz signal, there will be 46,000 partials in the modulated signal. Notwithstanding the tiny amplitudes of most of them, that should do the trick!

There's just one problem with this result. You should know from last month's discussion that a kettle drum produces just one partial below the principal, but my — albeit arbitrary — choice of 100Hz and 87Hz has generated about a dozen components below 150Hz. Increasing the frequencies of the signals feeding the ring modulator does not alleviate this, so I will place an high-pass filter in the signal path to remove the offending low frequencies (as shown in Figure 9 below). Now you have only to make the Decay/Release of this contour generator suitably short, and you have the desired burst of closely spaced, enharmonic modes that decay quickly after the membrane is struck.

| | | | FREQUENCIES OF FIRST FIVE 100HZ CARRIER HARMONICS | | | | |
|---|---|---|---|---|---|---|---|
| | | | 100 | 200 | 300 | 400 | 500 |
| FREQUENCIES OF FIRST FIVE 87HZ MODULATOR HARMONICS | 87 | $w_1+w_2$<br>$w_1$-$w_2$ | 187<br>13 | 287<br>113 | 387<br>213 | 487<br>313 | 587<br>413 |
| | 174 | $w_1+w_2$<br>$w_1$-$w_2$ | 274<br>74 | 374<br>26 | 474<br>126 | 574<br>226 | 674<br>326 |
| | 261 | $w_1+w_2$<br>$w_1$-$w_2$ | 361<br>161 | 461<br>61 | 561<br>39 | 661<br>139 | 761<br>239 |
| | 348 | $w_1+w_2$<br>$w_1$-$w_2$ | 448<br>248 | 548<br>148 | 648<br>48 | 748<br>52 | 848<br>152 |
| | 435 | $w_1+w_2$<br>$w_1$-$w_2$ | 535<br>335 | 635<br>235 | 735<br>135 | 835<br>35 | 935<br>65 |

Table 3: The first 25 partials generated by amplitude modulation of two signals with frequencies 100Hz and 87Hz.

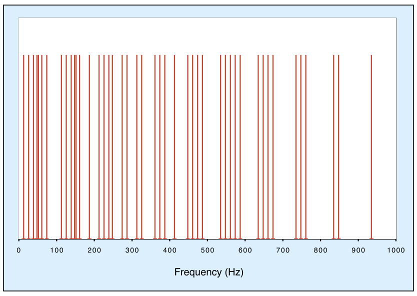


Figure 8: The 25 partials shown in Table 3.

At this point, I should make it clear that the partials generated by ring modulation do not lie at the correct frequencies for a kettle drum. Nonetheless, the effect should be

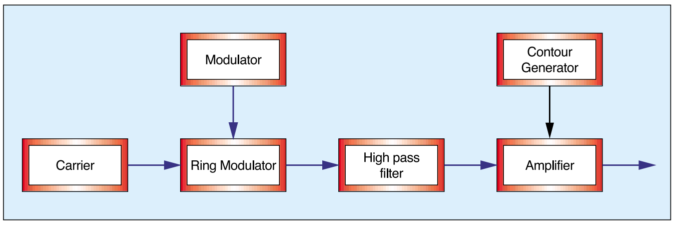


Figure 9: Eliminating the unwanted low frequencies in the quasi-noise burst.

sufficient for your purposes.

It's now time to consider the strength with which the drum's membrane is struck. As explained last month, this affects only the loudness of the sound, so you can emulate this by adding a VCA at the end of the signal chain, perhaps controlling its gain using keyboard velocity (see Figure 10, right).

Finally, the pitch of the sound must be controlled. You could use the keyboard to 'play' the sound in conventional semitones, but real timpani do not work this way. I therefore propose to use the pitch-bend wheel or a CV pedal to span a range of pitches (say, plus or minus a fifth) and restrict my playing to just one or two notes on the keyboard itself (see Figure 11, right).

You're now in a position to construct an analogue timpani synth, as shown in Figure 12 below. The eagle-eyed among you might notice that I've omitted the complex EQ bank that would follow the final amplifier to model the sound dispersion pattern of the real instrument, but I hope you'll forgive me that — it'll help to keep the graphics department at

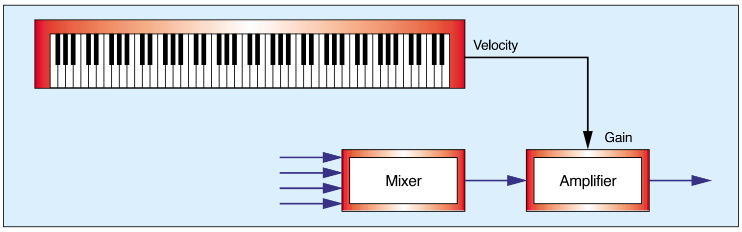


Figure 10: Controlling the loudness using keyboard velocity.

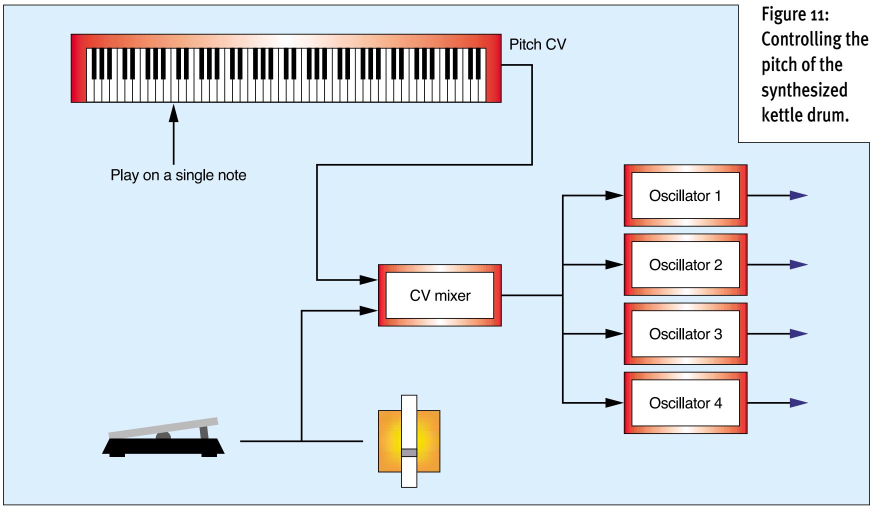


Figure 11: Controlling the pitch of the synthesized kettle drum.

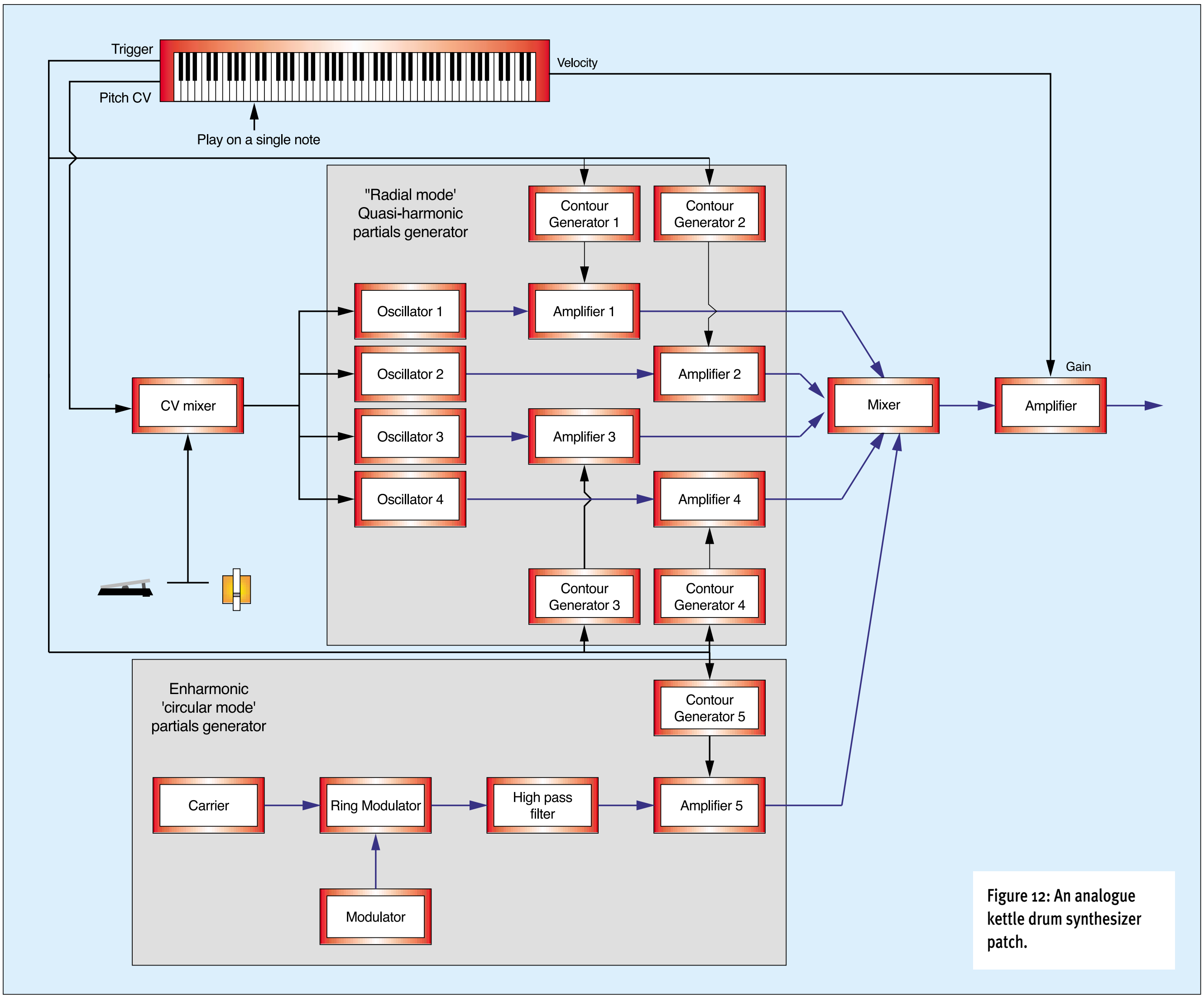


Figure 12: An analogue kettle drum synthesizer patch.

*Sound On Sound* on my side.

If Figure 12 looks complex, it isn't as bad as you might fear. The 'radial mode' generator comprises 12 modules, but these are just four sets of an oscillator/amplifier/contour generator setup. Likewise, the 'circular mode' generator, pitch controls and final gain are simply Figures 9, 10 and 11 incorporated into the patch.

## Sometimes Digital *Is* Best...

Unfortunately, most of us do not have access to the powerful modular analogue synth needed to create the patch in Figure 12. So, in a moment, I'll demonstrate how a simple and relatively cheap analogue synth can give you a surprisingly accurate imitation of the kettle drum. However, tens of thousands of us have an ideal synth for creating the patch. You may not realise it... but if you own a DX7 or any other six-operator FM synth, you're in business.

Look at algorithm 31 on a DX7... it's the same as Figure 13 below. Now, remembering that each operator on a DX7 is a sine-wave oscillator with an associated pitch envelope generator, amplitude envelope generator, and amplifier, it's clear that you can use Operators 1 to 4 as the 'radial mode' generators in Figure 12. Indeed, so suitable is the DX7 that you can type the 1.00:1.50:1.99:2.44 frequency ratios directly into the algorithm.

Similarly, Operators 5 and 6 are equivalent to the 'circular mode' generator in Figure 12. OK, so the DX7 algorithm uses FM to generate its enharmonic frequencies, whereas the analogue patch used a ring modulator, but the effect is much the same.

Photo: Richard Ecclestone

The Yamaha DX7.

Indeed, FM produces even more components with no harmonic relation to the pitched part of the sound, and is arguably even more suitable than ring modulation.

Using my DX1, I created an approximation to the patch in Figure 12 in a little under five minutes. Even in its rough form, it was remarkably usable.

I then loaded Yamaha's own timpani patch. Ooh... this uses algorithm 16, has just one carrier, and sounds far better than my first attempt. The difference appears to lie in the realism of the initial part of the sound so, if you want to program the best possible imitation, you'll need to analyse the short-lived enharmonic components more accurately. Unfortunately (or fortunately, depending upon your fondness for higher maths) a more detailed model of the sound is beyond the scope of Synth Secrets.

So let's now return to the analogue domain and see if it's possible to simplify the patch while still retaining the essence of the desired sound.

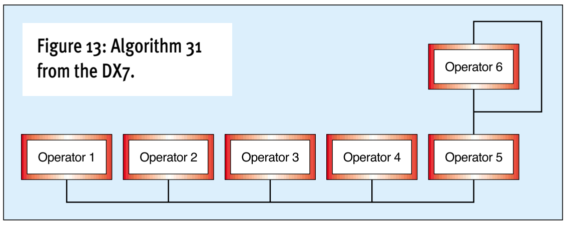


Figure 13: Algorithm 31 from the DX7.

## The Korg MS20

Figure 14, below, shows in simplified form the control panel of that most popular of vintage synths, the Korg MS20. Normally associated with the squeaks and squeals of modern dance music, this is an under-rated synth for careful, imitative synthesis. The reason for this generally lies in its rather lacklustre 12dB-per-octave filters, which are not ideal for powerful sounds such as brass. Nevertheless, the filters need not always be a limitation, as you will see.

Since the MS20 has just two oscillators, and neither of these produces a sine wave, you can't create the pitched part of Figure 12 as drawn. Nor can the MS20 produce slightly detuned partials.

So you must compromise... you need to choose a waveform with just a few strong harmonics, and filter off the rest of them. You can do this by choosing the triangle wave for Oscillator 1, ignoring Oscillator 2, and reducing the low-pass filter cutoff frequency (see Figure 15).

However, you can do better. If you refer back to Figure 6, you'll see that the relative

Korg's MS20.

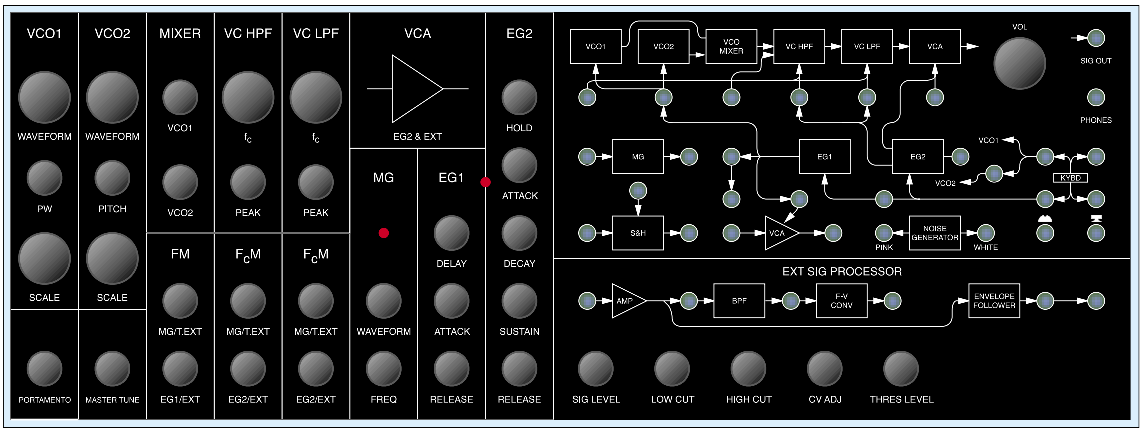


Figure 14: A simplified representation of the MS20 control panel.

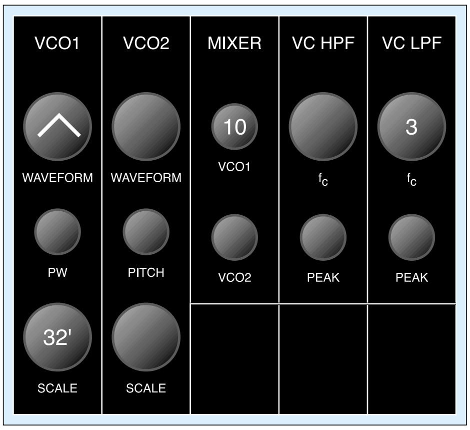


Figure 15: The oscillator setting for the kettle drum patch.

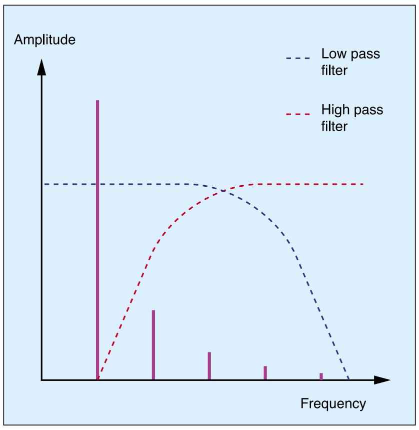


Figure 16: Applying high-pass and low-pass filters to sculpt the harmonic content of the sound.

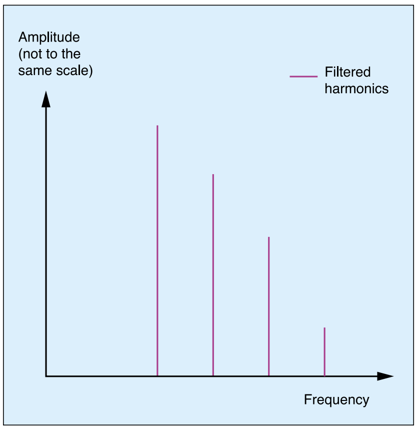


Figure 17: The output from the filters defined in Figure 16.

amplitudes of the first three partials are only slightly different. So you can use the MS20's twin filters to sculpt the '1/n²' amplitude relationship of the triangle wave (shown in Figure 16) into something far more akin to the desired shape (see Figure 17). Notice in particular how this removes the fundamental of the triangle wave, and ensures that the first four partials have the required 1:1.5:2:2.5 relationship seen in Table 1, rather than 1:2:3:4, as would be the case if the fundamental survived filtering.

You can improve this even further by adding resonance (or 'peak' in Korg-speak) to the filters' cutoff frequencies, further enhancing the contributions from the partials contained in the narrow band of frequencies between the two cutoff frequencies.

But what of the non-pitched partials in the kettle drum sound? You can use Oscillator 2 to provide these if you select the Ring Modulator option and the lowest footage, and then detune the oscillator so that it produces a suitably clangorous tone (I find that something in the region of +2 works quite well). You can then mix a little of this with Oscillator 1 to create the desired timbre.

You now create a simple Attack/Decay/Release envelope using EG2 to shape the amplitude of the note. Remember to set the decay and release times to be equal, so that the sound decays consistently whether you release the key or not. Finally, feed a little of the CV from EG2 to the low-pass filter cutoff frequency. This makes the sound brighter at the start, and reduces the brightness as the volume decays. This is correct: remember that the higher partials in Figures 1, 2 and 3 decayed more rapidly than some of the lower ones, so applying the EG to the low-pass filter overcomes the lack of independent VCAs. The resulting patch appears in Figure 18 (below left).

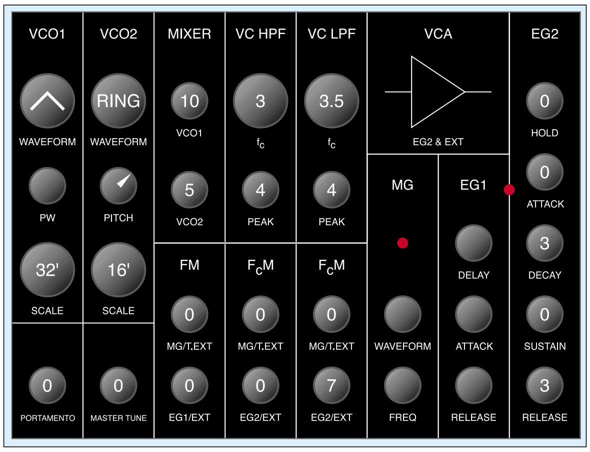


Figure 18: A simple kettle drum patch for the Korg MS20.

## A Better MS20 Patch

The patch in Figure 18 is very sensitive to tiny changes in certain parameters — particularly the VCO2 pitch and the filter settings — but once set correctly, it can give the impression of a kettle drum. Nonetheless, it's not particularly satisfying. Indeed, considering the care with which I've followed the theory, it's remarkable just how unconvincing it is. I suspect that this is due to the limitations of the MS20's ring modulator, which gives no control over the input signals, so let's cast theory to the wind and see whether another method can

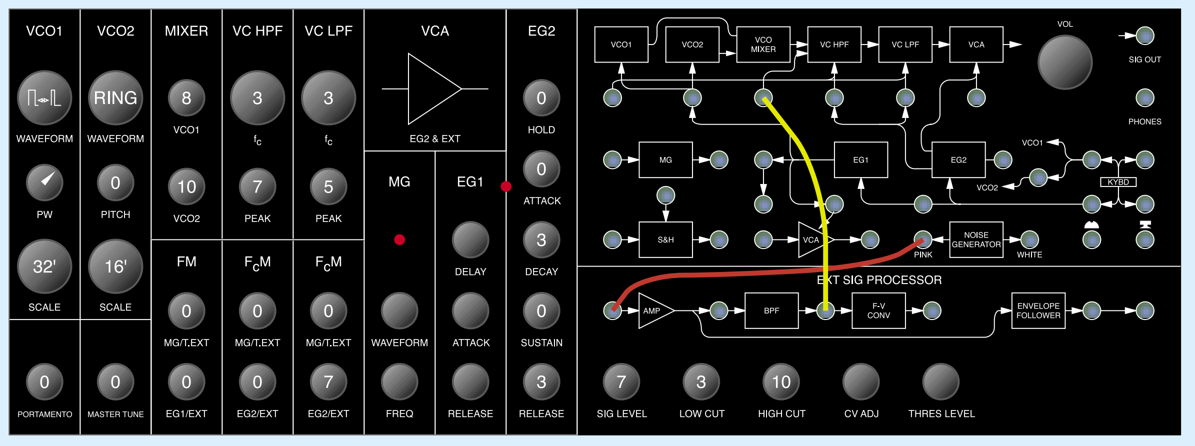


Figure 19: A more complex kettle drum patch for the Korg MS20.

produce more convincing results.

If you study Figure 19, you'll see that my next patch (which is Korg's 'factory' timpani patch) uses a pulse wave as its starting point, and that the ring modulator is not detuned. This makes the basic timbre much less clangorous than the patch in Figure 18. Other parts of Figure 19 are more familiar: both filters again have resonance, further emphasising the second, third and fourth harmonics, and EG2 again affects the low-pass filter cutoff frequency.

However, if you play this part of the patch alone, you'll find that it sounds nothing like a kettle drum... it's just a simple bass sound, and not a very good one, at that. But now have a look at the semi-modular patch matrix provided by the MS20, shown on the right-hand side of the diagram.

Look closely, and you'll see that the noise generator is feeding pink noise to the input of the External Signal Processor. This type of noise is already deficient in high frequencies, and the high-pass filter in the ESP is removing the low frequencies, so the result is a narrow band of noise. Furthermore, at the settings shown, the input of the ESP is distorting, so the result is a very 'rough' timbre. This is then directed to the input of the MS20's main filters, where it is even further narrowed and emphasised by the overlapping resonant filters.

The result is a band of highly tuned, somewhat distorted noise that takes the place of the ring-modulated sound in Figure 12, and does so very effectively. Indeed, it sounds remarkably similar to the unpitched sound produced by the kettle drum. If you now add back the pitched partials from the oscillators, the result is extraordinarily life-like, with the sound from the oscillators providing the characteristic 'ringing' of the kettle shell as the sound decays.

Having got this far, you could go on to patch the CV Wheel to the oscillator pitches (as suggested in Figure 12) and develop a number of small performance enhancements. However, lack of space precludes me from doing so here, and this is something that you'll just have to try for yourself.

## More Patching Ideas

If you look back to the start of this article, you'll see that I've gone full circle. Having admitted that, in Figure 1, the 'noise' representation of the enharmonic modes is the result of my laziness, it turns out that using a noise generator is indeed a highly satisfactory way to model a kettle drum. But why stop there...?

As I discussed last month, timpani are far from the only pitched membranophones so, if you own an MS20, you should consider using these ideas to create a wide range of pitched drum sounds. This is simply done: take the patches in Figures 18 and 19, and experiment with the pitches, filter settings and envelope times. You should easily create other pitched percussion sounds. However, you must take care not to stray too far from the initial settings; do so, and the character of your sounds will change into something else entirely.

Of course, you can try all of this on other synthesizers, but among low-cost instruments, the MS20 is particularly suitable because it offers a powerful combination of pitched oscillators, a ring modulator, a noise source, and four filters. Consequently, you won't get the same quality of results from the Minimoog, ARP Axxe or Roland SH101 that I used so successfully for brass patches a few months ago.

So there you have it... proof that the MS20 is good for far more than dancefloor bleeps and bloops. Give it a try! SOS

Gordon Reid

# Synth Secrets

**Ever wanted to synthesize unpitched membranophones? No? Well, you might if you knew that bass and snare drums are of this percussion type. We show you how...**

## Synthesizing Drums: The Bass Drum

Two months ago, I introduced into Synth Secrets the idea of membranophones, the family of instruments that includes all the common drums, and analysed the sonic characteristics of the pitched subset of that family. We then proceeded to apply this knowledge to the synthesis of timpani.

Enlightening though the process of synthesizing timpani is, I realise that you're more likely to be interested in synthesizing the types of drums used in modern music — bass drums, snare drums, and toms — all of which fall into the subset of *unpitched* membranophones. So this month, we're going to start looking into the analogue synthesis of common drum sounds.

We'll start by casting our mind back to the kettle drum. If you remember my analysis of two months ago, you'll know that the environment in which a membrane finds itself influences its modes of vibration. I covered some of this ground last month, but let's now go into a little more detail.

Figure 1(a) shows the idealised case of a circular membrane suspended in a vacuum. As we already know, it is this that produces the enharmonic set of frequencies determined by the Bessel Functions. Of course, it's highly unlikely that either you or I will ever experience this sound. Firstly, no sound is carried in a vacuum. Secondly, much as I would like to eject most of my band's former drummers from the airlock of a passing Vogon Constructor Vessel, the opportunity to do so has not yet arisen. Therefore, we must consider the vibration of the membrane when suspended in the atmosphere, as shown in Figure 1(b).

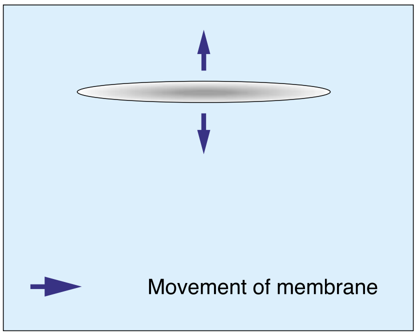


Figure 1(a): A membrane suspended in a vacuum.

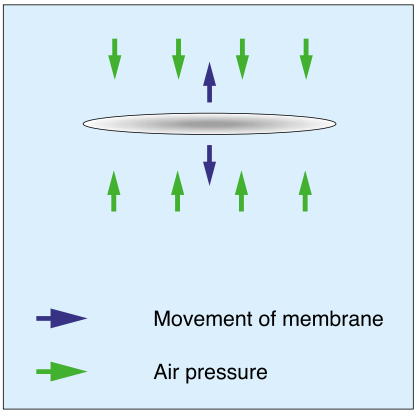


Figure 1(b): A membrane suspended in air.

As you can appreciate, it's not a trivial task for the membrane to shift all the air adjacent to its surfaces, especially if it is vibrating at scores — if not hundreds — of Hertz. And the result of this effort is that the frequencies of the modes shift upwards, and that they become somewhat less enharmonic.

Despite the ease with which we can create this sound, it is still not one that you will often hear. No — the membrane sound we all know and love is the one produced when you stretch the membrane and mount it on some sort of shell.

The closed shell of the kettle drum discussed last month is a particular example of this, with the vibrations of the membrane strongly influenced by the modes of vibration of the air inside the shell itself. Nonetheless, as shown in Figure 1(c), the membrane still has to shift the air on its outer surface against atmospheric pressure.

The physics of this example is, as we have already discussed, rather complex, but we know that the consequences of mounting the membrane in this way are twofold. Firstly, the pitches of the modes rise even further. Secondly, the frequencies of the important radial modes become almost harmonic in their distribution.

OK, so far we've discussed nothing new. But let's now take a step backwards, and ask what happens when we remove the body of the kettle drum and replace it with a tubular shell capped by a *second* movable membrane (see Figure 2, above right). Clearly, we have designed a conventional drum of one sort or another. Welcome to the world of the unpitched membranophones.

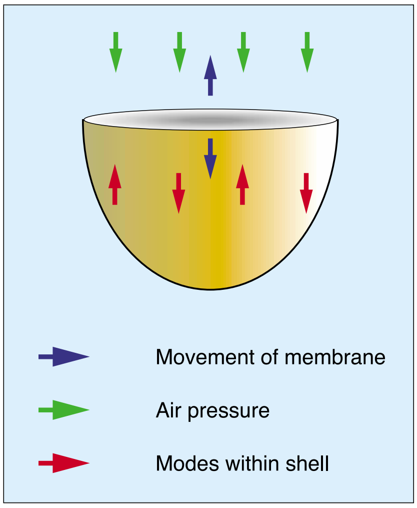


Figure 1(c): A membrane mounted on a rigid, fully enclosing shell.

## Bass Drums — Frequency Content

Traditional bass drums are much as shown in Figure 2. The head that you strike is the beating head or batter head, while the other is the resonating head or carry head. Clearly, if the batter and carry are without holes, the air in the drum is trapped, and the membranes' modes of vibration will again be influenced by the cavity modes. However, whereas the modes within the rigid kettle drum shell are easy to understand and calculate, those of the bass drum are not. This is because, whereas the kettle drum has two coupled resonators (the membrane and the cavity) the conventional drum, with its second membrane, has three coupled resonators.

The physics of this is too advanced for Synth Secrets, but we can understand the measurements of its behaviour observed by academics. Let's start by considering Table 1, which shows a set of bass drum modes when both membranes are stretched to the same tension.

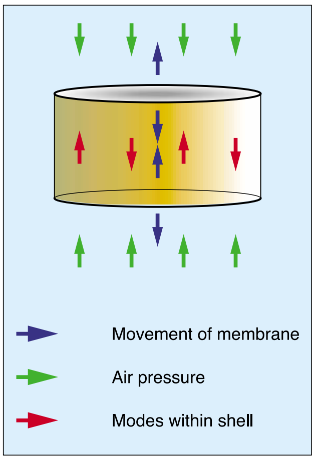


Figure 2: A membrane suspended on a shell capped by a second membrane.

First, I had better explain why there are two frequencies shown for the 0,1 and 1,1 modes. Again, without describing the physics involved, we observe that the membranes' vibrations are affected by the air between them in such a way that, for these modes, not one, but two, frequencies are produced. Weird, eh?

| MODE | FREQUENCY (Hz) |
|---|---|
| 0,1 | 50, 118 |
| 1,1 | 86, 93 |
| 2,1 | 136 |
| 3,1 | 182 |
| 4,1 | 225 |
| 5,1 | 273 |

Table 1: The modal frequencies of a dual-membrane bass drum.

Now, the frequencies in Table 1 may not seem related in any way, but when we plot them on a chart, something unexpected happens. If you look at Figure 3, the frequencies may look enharmonic, but if we remove the doubled 0,1 mode at 118Hz and the doubled 1,1 mode at 86Hz, we obtain Figure 4.

Yikes! I don't know how it looks to you, but to me Figure 4

▶

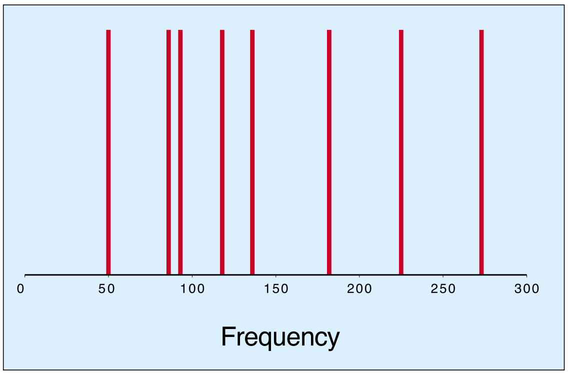


Figure 3: Frequencies of the modes of a bass drum.

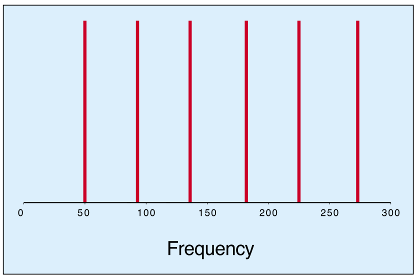


Figure 4: A simplified plot of the bass drum modes.

looks very close to a harmonic oscillator. This is, however, an illusion (damn... just when you thought you were getting the hang of things!). Although the spacing of the modes is very regular (see Table 2 below), the frequencies of the upper partials do not lie at integer multiples of the lowest component frequency. In fact, they appear to be in the form of a harmonic series, frequency-shifted upwards by about 7Hz. I can demonstrate this by subtracting 7Hz from each of the frequencies in Table 2, and writing the results in Table 3... which gives us an idea about how to synthesize the spectrum in Table 2.

Let's take a complex harmonic waveform with a fundamental frequency of 43Hz, and pass its output through a device called a frequency-shifter (see the box on the next page). If we use this to increase by 7Hz the frequencies of each component in the signal (ie. at 43Hz, 86Hz, 129Hz, 172Hz... and so on), we obtain a signal with partials at 50Hz, 93Hz, 136Hz and 179Hz... which is almost exactly the spectrum we require.

Of course, we have no idea of the relative amplitudes of the partials in the sound, nor any idea about the relative speeds at which they decay. Nevertheless, we could pass this signal through a VCA controlled by an AR contour generator and — if we are very lucky — we might obtain something that sounds like a bass drum (see Figure 5 below).

| FREQUENCY (Hz) | DIFFERENCE (Hz) | HARMONIC RATIO |
|---|---|---|
| 50 | | 1.00 |
| 93 | +43 | 1.86 |
| 136 | +43 | 2.72 |
| 182 | +46 | 3.64 |
| 225 | +43 | 4.50 |
| 273 | +48 | 5.46 |

Table 2: The modal series shown in Figure 4.

| FREQUENCIES FROM TABLE 2 MINUS 7Hz (Hz) | HARMONIC RATIO |
|---|---|
| 43 | 1 |
| 86 | 2 |
| 129 | 3 |
| 175 | 4.07 |
| 218 | 5.07 |
| 266 | 6.19 |

Table 3: The modified series is almost precisely harmonic.

Unfortunately, these low, quasi-harmonic modes are not the only important components in the sound of the drum. Because our ears are highly attuned to the frequency range that contains human speech, we are particularly sensitive to partials in the region of a few hundred Hertz. Inevitably (because the universe hates us and wouldn't want anything to be easy), the drum generates scores of partials between 250Hz and 1kHz, so we must also synthesize these. This isn't a problem, because we can recreate the required cluster of enharmonic modes using frequency modulation (FM) synthesis. We simply add to Figure 5 two oscillators — one modulating the other — plus a band-pass filter (see Figure 6).

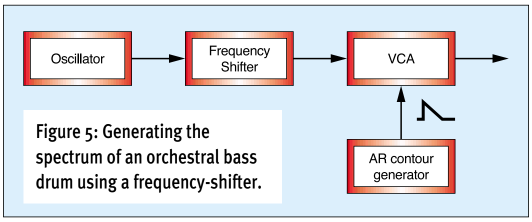


Figure 5: Generating the spectrum of an orchestral bass drum using a frequency-shifter.

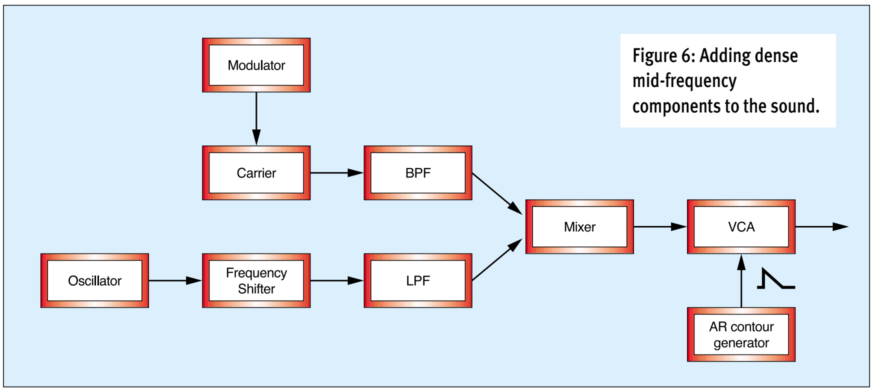


Figure 6: Adding dense mid-frequency components to the sound.

But before we get carried away, analysing and developing this model still further, let's take a step backwards for a moment. We started by assuming that the bass drum had two membranes of equal tension, and was completely sealed at both ends. This is seldom true, because players of orchestral drums rarely tune their instruments like this, tending to tune the carry head at a much lower tension.

Table 4 (above) shows the measurements for a drum tuned in this fashion. As you can see, the dual frequencies previously exhibited by the 0,1 and 1,1 modes have disappeared. But, more important than this, the result is close to a true harmonic series without any offset. This means that, when we synthesize the sound of a drum with a detuned membrane, we can dispose of the frequency-shifter.

Of course, the bass drums we encounter in most modern music do not have two complete membranes. We call them kick drums and they often have a hole cut in the carry head, into which we stuff pillows, microphones, the guitarist's head, and the occasional empty lager can (see Figure 7).

| MODE | FREQUENCY (Hz) | HARMONIC RATIO RELATIVE TO 46Hz |
|---|---|---|
| 0,1 | 44 | 0.96 |
| 1,1 | 91 | 1.98 |
| 2,1 | 138 | 3.00 |
| 3,1 | 184 | 4.00 |
| 4,1 | 232 | 5.04 |
| 5,1 | 282 | 6.13 |

Table 4: The modal frequencies of a dual-membrane bass drum with the carry head detuned with respect to the batter head.

We might expect that this would introduce yet another dose of arcane concepts and hideous maths, but — for once — the result is fairly straightforward. The action of the atmospheric pressure at the aperture of the partial membrane is not so different from the loose (ie. detuned) membrane, so the frequencies in Table 4 are largely unaffected. This then leads us to an important conclusion:

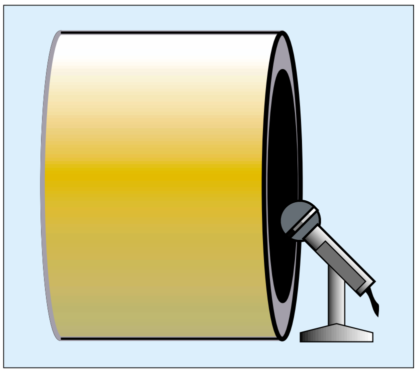


Figure 7: A bass drum with a hole in the carry membrane.

*The frequency components of a kick drum approximate a harmonic spectrum at low frequencies, with a large number of densely packed enharmonic components at mid and high frequencies.*

## Bass Drums — Decays, Pitch Changes & Clicks

Now that we know the frequency content of the bass drum, we need to know more about the relative amplitudes of the components, and the rates at which they decay. There is very little academic literature about the first of these, but I think we can assume that the amplitudes of the lowest modes are the loudest, and that these amplitudes diminish rapidly with increasing frequency. If so, a triangle wave or a filtered sawtooth wave will be an ideal starting point for bass-drum synthesis. As for the decay rates, a number of texts suggest that the rate is constant across all the important partials. This means that — although I was guessing when I drew Figures 5 and 6 — we should be safe using a single VCA and EG to simulate the decay of the sound.

It might seem that we now know everything we need to know in order to create a bass drum patch, but there are two important attributes left to investigate: first, the pitch shift that occurs every time you excite the batter head; and second, the sound of the beater 'click'. Let's look at the first of these.

If you again cast your mind back a couple of months, you may recall that the cavity modes within a kettle-drum shell act as frequency regulators on the equivalent

## What Is A Frequency-shifter?

I introduced many synthesizer modules throughout parts one to 23 of this series. I started with obvious ones such as oscillators and filters, and then proceeded to discuss some of the more obscure ones — envelope followers, sample & hold modules, keyboard decoders, and so on. However, I didn't mention them all, and one of my omissions was the Frequency-shifter — often called the Bode Frequency-shifter, in honour of the man who developed it for the early Moog modular synthesizers.

The Analogue Systems RS240 Frequency Shifter.

Unlike a Harmoniser or pitch-shifter, which alters the frequency of each component in the sound spectrum by a fixed ratio, a frequency-shifter moves each of the components by a fixed amount in Hertz. To understand the difference, we'll start by considering the first of the diagrams on the right. This shows the spectrum of a 100Hz signal with four harmonics.

If we pass this signal through a pitch-shifter set up to increase the pitch of the fundamental by one octave, we obtain the spectrum shown in the second diagram. And, since we have doubled the frequency of the fundamental, all the harmonics have moved too: the second harmonic still lies at twice the fundamental frequency, the third harmonic still lies at three times the fundamental frequency... and so on.

But what happens if we build a device that moves the frequencies of each component of the sound by an equal amount — say, 100Hz — rather than by an equal ratio? This is the very essence of a frequency-shifter.

As you can see from the third diagram, the signal still *looks* harmonic, but it is not. The first overtone (it's no longer a harmonic) lies at 300Hz, which is 1.5 times the lowest frequency, the second overtone lies at twice the lowest frequency, and the third overtone lies at 2.5 times the lowest frequency.

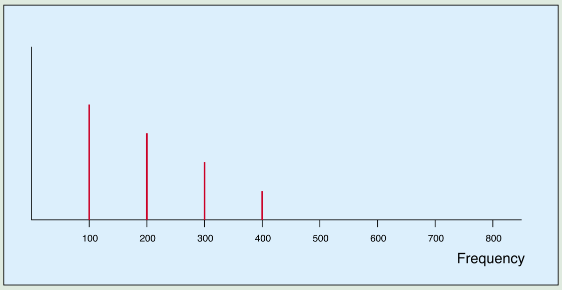
In this particular case, the signal shown will still sound musical, because the components lie in a harmonic pattern based on a frequency of 100Hz; it's just that the fundamental is missing. Indeed, the human brain is such an amazing piece of equipment that, if the signal components lie in precisely the right places, it will insert the missing pitch, and you will 'hear' a note of 100Hz, even though the lowest partial lies at 200Hz. The builders of church and cathedral organs have been using this trick for centuries, tuning the pipes so that you hear implied fundamentals lying an octave below the true pitch. Given that organ pipes in even moderate organs extend to 16', and that the implied pitch is one that would emanate from a pipe 32' long, you can appreciate that this represents a huge saving in height and weight.

Let's now extend this argument. Instead of doubling the frequency of the lowest frequency component, I'll increase it by 25 percent, as in the last diagram.

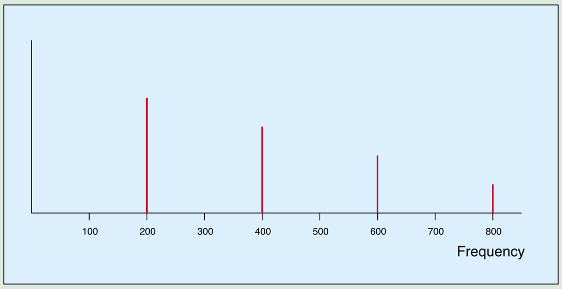
If we pass the signal through a pitch-shifter, the fundamental moves from 100Hz to 125Hz, and the overtones are likewise increased in frequency by 25 percent to 250Hz, 375Hz, and 500Hz. In practice, this means that the pitch of the note has shifted upwards by about two semitones.

But if we use a frequency-shifter to increase the frequency of the fundamental by 25Hz, the overtones also move by 25Hz, to lie at 225Hz, 325Hz and 425Hz. This means that the overtones lie at 1.8 times, 2.6 times, and 3.4 times the lowest component. Clearly, the signal is no longer harmonic, and it will not sound like a tuned musical tone.

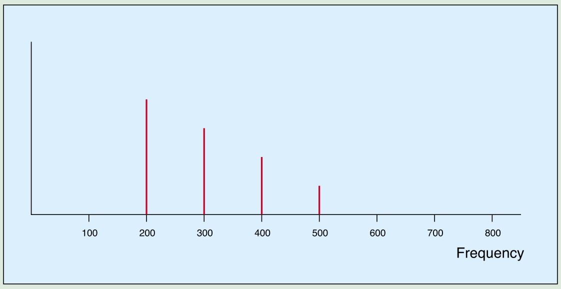
As you might imagine, there are numerous uses for such a device, and the frequency-shifter can generate a wide range of effects. These include eerie chorus and phasing effects, ring modulation effects, stereo panning, Doppler effects, and more. You can even use them to reduce acoustic feedback. But these are all topics for another day.



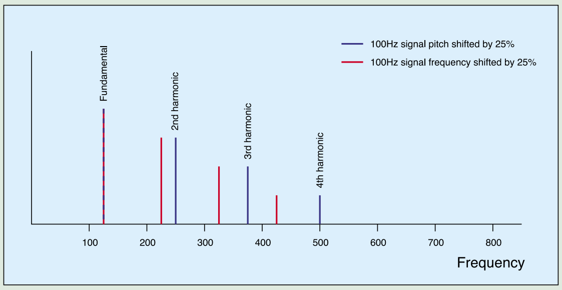
A 100Hz signal with four harmonics.



Pitch-shifting the fundamental from the first diagram by 100Hz (ie. moving the note up one octave).



Frequency-shifting the original signal by 100Hz.



Comparing the results of pitch-shifting and frequency-shifting.

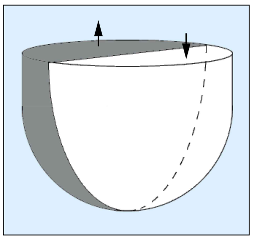


Figure 8: Coupling of the membrane and the internal modes of a kettle drum.

vibrations of the membrane. Therefore, no matter how hard or softly you hit the membrane, the frequencies of its modes remain tightly locked to those permitted by the air in the cavity. Figure 8 depicts the 0,1 mode of a kettle drum, showing how the vibration of the air is coupled to the vibration of the membrane.

Now let's look again at the bass drum in Figure 7. This has a flexible interface with the outside world (ie. the air section at the aperture in the carry head), so the cavity modes are very much less constraining that those of the kettle drum. This means that, if the batter head wants to vibrate at a different pitch, it is relatively free to do so. But what would make it change pitch?

Consider Figure 9(a). This shows a stretched membrane seen from its edge. I have arbitrarily made it 30 inches in diameter.

Now let's beat the living daylights out of this, smacking it with a beater and displacing its centre by an inch or so. This is an unrealistically large displacement for any tightly stretched membrane, yet it only increases the distance across the surface by about one sixteenth of an inch, as shown in Figure 9(b) below.

What it does do, however, is increase the tension of the membrane by an amount that is proportional to the square of the displacement. And, since pitch is determined by tension, this increases the pitches of the modes by a considerable amount. It's a small leap of understanding to realise, therefore, that the pitch of every mode will be higher at the start of the sound (when the maximum instantaneous displacement is large) and will drop as the amplitude of the vibration decays. Indeed, the pitch of a typical kick drum can shift by a couple of semitones from start to finish, and we must build this into our patch if it is to sound realistic.

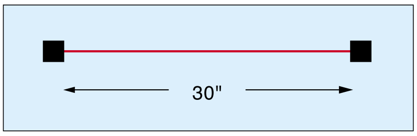


Figure 9(a): A membrane at rest.

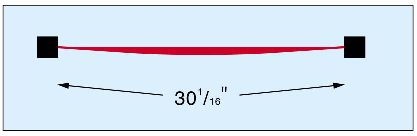


Figure 9(b): A struck membrane.

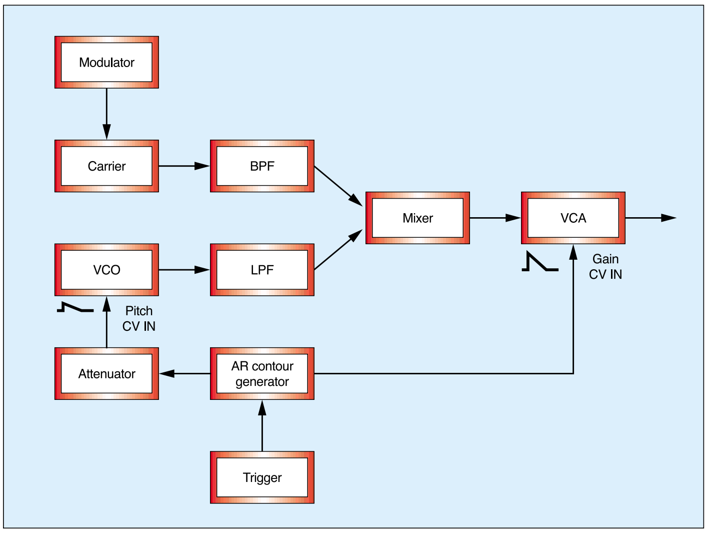


Figure 10: Using a single contour generator to modify loudness and pitch.

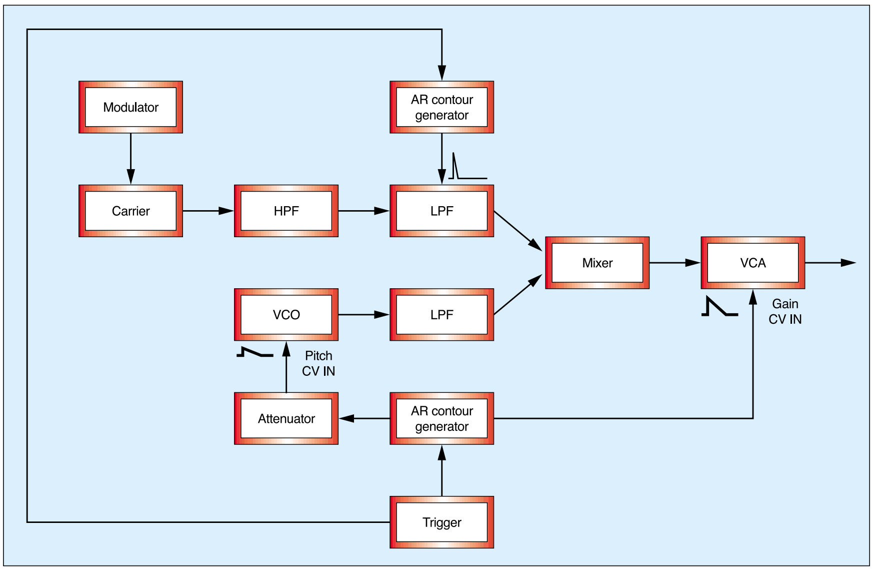


Figure 11: A bass drum patch with pitch-bend and click.

Fortunately, we do not need a second contour generator to implement this. After all, the loudness and the pitch of the sound are both determined by the maximum instantaneous displacement of the membrane, so a single AR Generator should do the trick. However, whereas the VCA Gain will change by 100 percent from the start to the end of the sound, the pitch should only shift by around 10 percent, so the patch requires some form of attenuator at the oscillator's pitch CV input (see Figure 10 above).

Finally, we come to the beater click. This is caused by hundreds of short-lived high-frequency partials that exist for just a few milliseconds after the membrane is hit.

There are two ways we can model this. One would be to use a short noise burst (which you would normally call a click); the other uses a contour generator to shape the spectrum of the partials generated by the FM components in Figure 10. If we choose the latter, we split the band-pass filter into its low-pass and high-pass components, and apply a rapid AR contour to the cutoff frequency of the LPF. This allows many high-frequency components (almost a noise spectrum) to pass for a very brief time, before the patch settles down to the sound generated in Figure 10 (see Figure 11 above).

So there we have it: a simple bass-drum patch. Simple? Well, we've skirted over the true nature of the enharmonic partials, approximated the decay rates, disregarded the (albeit reduced) effects of the cavity modes, and totally ignored the presence of any shell resonances. Consequently, I think that it's fair to say that this is a simplified patch. Nonetheless, it will produce extremely usable results. If you have access to a patchable analogue synth with three oscillators, cross-modulation, three suitable filters, a couple of contour generators, a mixer and a couple of VCAs, you're in business.

Unfortunately, few low-cost analogue monosynths have this degree of flexibility. So next month we'll set about programming some bass drum sounds on the synths you *do* own. Until then... SOS

# Synth Secrets

## Practical Bass Drum Synthesis

Moving from last month's theoretical bass drum synth patch to its practical application on affordable analogue synths, we also take a look at how the world's most famous drum machines produce this fundamental rhythm sound...

Gordon Reid

Last month, we analysed the bass drum, ending up with a patch that synthesized all the important elements of its sound (see Figure 1(a), right). Given a synthesizer of appropriate power and flexibility, there's no reason why you shouldn't use this to create a wide range of powerful kick drum sounds. Indeed, with careful choices of VCO pitches and waveforms, filter characteristics, and contour rates, you can use this to synthesize a huge range of realistic and electronic percussion instruments.

Unfortunately, few of us have access to a synth capable of reproducing this patch, so I'm going to simplify it by replacing the FM generator in the upper-left corner with a noise source. When filtered appropriately, this will simulate the dense cluster of mid-frequency partials that the FM section generated. If I also assume that the pitch CV input on the VCO will have some form of level control, I can dispense with the attenuator and redraw the patch as Figure 1(b), shown right.

Unfortunately, simplified though this is, it's still too complex for the majority of analogue synthesizers. A Minimoog can't reproduce it, neither can a Roland SH101, an ARP Axxe, a Sequential Circuits Prophet 5, a Memorymoog, or even the semi-modular Korg MS20 and ARP 2600. But it's well-known that people do coax superb kick drum sounds from all of these synths — so how do they do it? That's what we're going to find out this month.

### Bass Drums On The Arp Axxe

The ARP Axxe is perhaps the simplest of the synths named above, offering just a single oscillator, a single LFO, a single filter, and a single contour generator. I have drawn the block diagram for this synth in Figure 2.

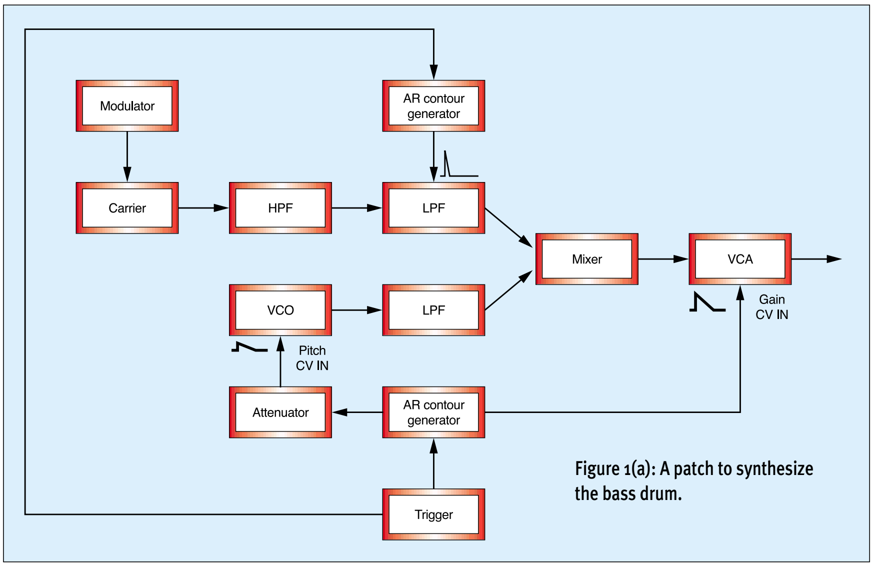


Figure 1(a): A patch to synthesize the bass drum.

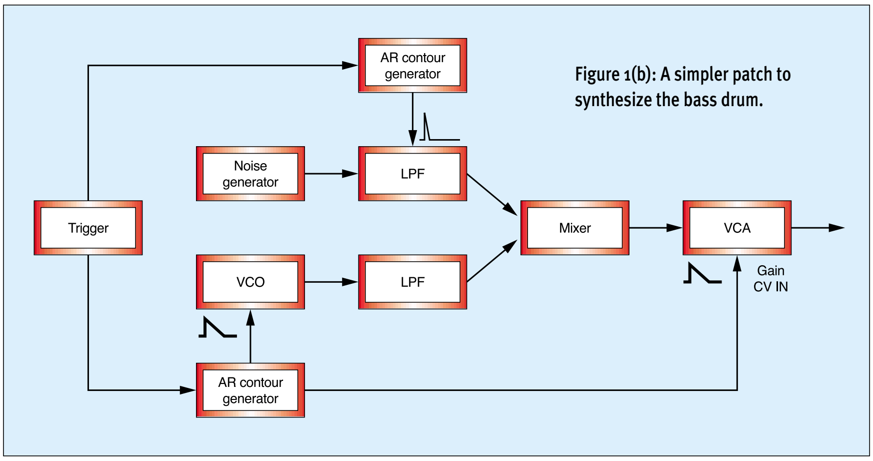


Figure 1(b): A simpler patch to synthesize the bass drum.

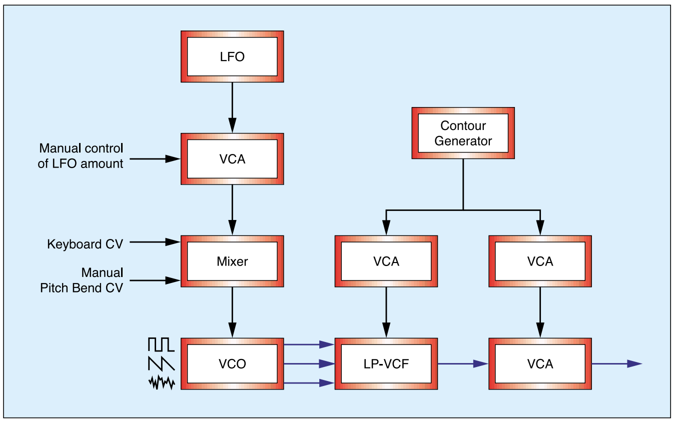


Figure 2: A block diagram of the important elements of the ARP Axxe.

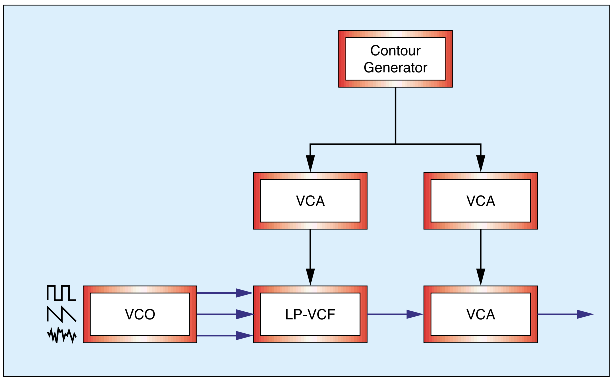


Figure 3: The elements used in the ARP Axxe bass drum patch.

Clearly, this lacks some of the modules present in both Figures 1(a) and 1(b) so, if we're going to be successful programming a bass drum on this, we're going to have to cut a few corners.

Let's start by getting rid of the things that we don't need. Nowhere last month did we discuss low-frequency modulation, and the Axxe's LFO won't oscillate at audio frequencies, so we can discard this and its associated VCA. Likewise, it's unlikely that we'll need pitch-bend, and we don't need to consider the keyboard CV, so these and their mixer can also disappear into the ether. That leaves a classic VCO/VCF/VCA signal path shaped by a single contour generator and two VCAs (see Figure 3, above).

Comparing this to Figure 1(b) — let alone 1(a) — you might imagine that there's little we can do to recreate the kick drum sound we want. But don't give up... all is not lost.

The most important thing that Figure 3 lacks is the ability to sweep the pitch of the VCO. However, the Axxe does offer a way to generate a tone with a downward sweep; we program this by setting the filter resonance to maximum, and using the contour generator to control the cutoff frequency. Since the Axxe's filter oscillates at high resonances, it produces something akin to a sine wave, and it is this that will sweep down in pitch as the contour progresses. So, knowing the contour we require (see Figure 4 below for a reminder) we can now set up the ADSR and VCF sections on the Axxe's front panel, as shown in Figure 5, above right.

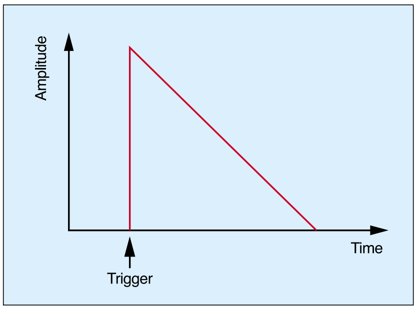


Figure 4: The pitch contour for the kick drum sound.

Scanning Figure 5 from left to right, you can see that the filter resonance is at its maximum, and that the amount of ADSR sweep applied is approximately 50 percent. There's no theoretical reason for choosing 50 percent... it's simply the setting that sounds good to me. You should also note that the filter's initial cutoff frequency is at its minimum, ensuring that the sweep ends at a very low frequency. For the same reason, there's no keyboard CV applied, and the LFO-to-VCF control is set to zero because we do not want the frequency to 'wobble' as it sweeps downwards.

Looking at the ADSR contour generator itself, you can see that (for reasons that I hope are obvious) the Attack time is zero. The Decay and Release times of approximately 50 percent are again empirical — they sound right to me — and they're the same length, so that the sound is consistent whether you keep the key pressed or not. Finally, the Sustain level is zero because the pitch of the drum always swoops down; it never 'sustains' at a single pitch.

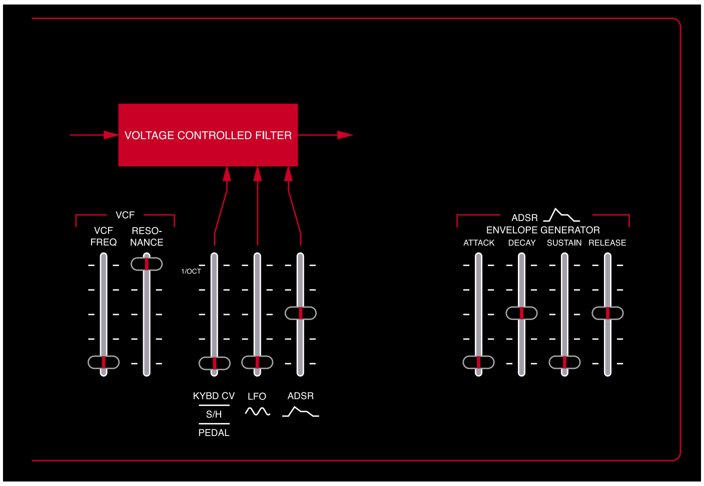


Figure 5: The VCF and ADSR settings for the Axxe kick drum sound.

Now, what about the amplitude curve of the sound? Last month I explained that that the pitch contour and the amplitude contour are likely to be the same, differing only in amount. This is very fortunate because, if it were not so, we would require a second contour generator, and the Axxe does not possess this. We therefore define the VCA response using the settings in Figure 6 (right). As you can see, we apply the full range of the ADSR to the VCA, and there is no initial VCA Gain, because this would stop the note decaying to silence.

Now for the cluster of middle frequencies in the kick drum sound. If we possessed more oscillators and signal paths, we could generate these in keeping with the scheme in Figure 1(a). But we don't, so we can't. The best we can do it to add the sawtooth and/or square waves plus some noise to the audio input of the filter. In theory, the saw and square add more harmonic content to the sound, and the noise adds something to emulate the cluster of enharmonic modes in the mid and upper frequencies. However, little of the signal presented to the input survives the path through the filter, so the output is largely unaffected by these signals.

We're now in a position to complete the Axxe kick drum patch (see Figure 7). Apart from what I've already mentioned, the only other thing to notice is that the Transpose switch is set to '-2 Octaves' so that the frequencies of the sawtooth and square waves are in the lowest register. And that's it... a beautifully simple and elegant patch which produces the classic 'analogue' kick drum sound. What's more, you can make it sound ▶

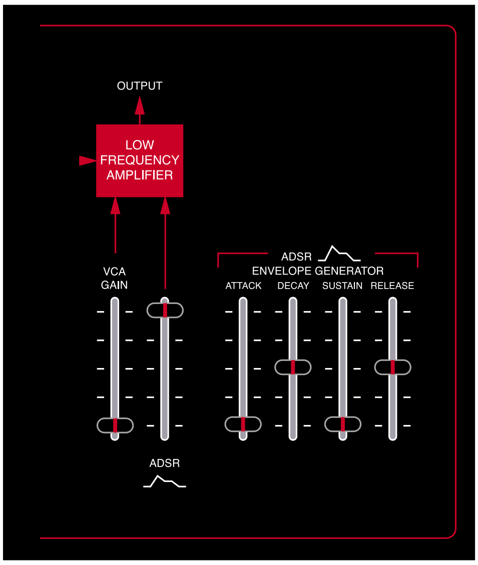


Figure 6: The Axxe's VCA and ADSR settings.

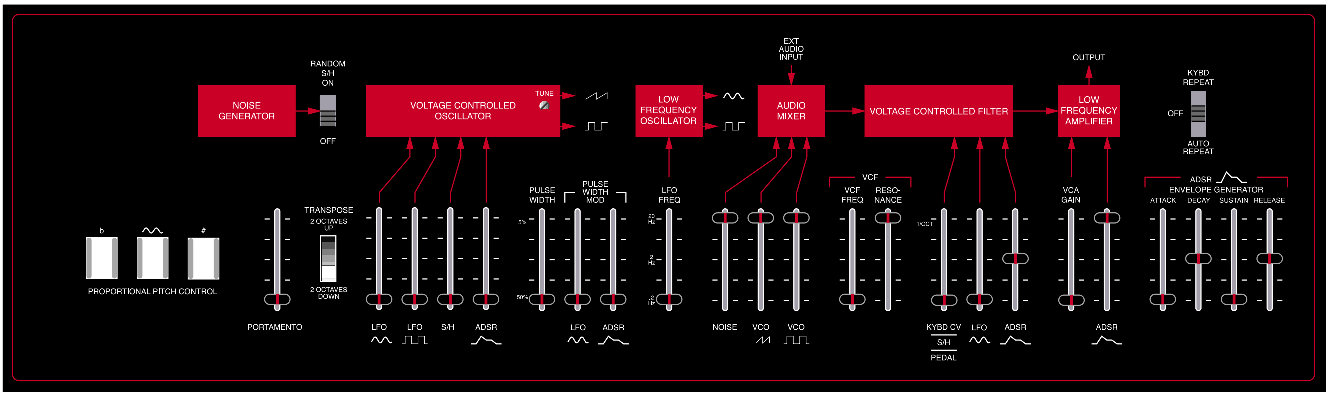


Figure 7: The Axxe kick drum patch.

more realistic (ie. like a real drum, not a beatbox) by reducing the filter resonance a tad, thus allowing a little of the VCO through the VCF. This adds a subtle tonal quality that sounds much like the shell of the kick drum. Then, if you set the trigger to 'Auto Repeat', and the LFO frequency to 2Hz, you have a killer 120bpm beat that's every bit as usable as those generated by dedicated drum machines.

## Bass Drums On The Roland SH101

The Roland SH101 offers a similar architecture to the Axxe, so it's not surprising that its kick drum patch is almost identical to the one we have just discussed. Sure, there are differences in the values of the settings, but these are merely a consequence of the different circuitry used. The philosophy of the patch is identical, and it yields very similar results.

So, we again set the VCF resonance to maximum, the cutoff frequency to minimum, and (in this case) the 'VCF Env' amount to about 60 percent. Likewise, the Decay and Release settings are in the region of '6'... and provided that all the other sliders are at zero, that's all there is to it, as shown in Figure 8 (right). If anything, this patch has more punch and depth than the Axxe's. This shouldn't be too surprising; ARP synths are renowned for their bright and fizzy sounds, not for warmth and thickness.

Tweaking Figure 8 yields numerous other bass drum sounds. For example, you can introduce some 16' sawtooth and/or the sub-oscillator to introduce a tonal quality to the sound. To do this, you increase the appropriate faders in the mixer, then reduce the 'VCF Env' amount and raise the cutoff frequency just far enough to let the lowest harmonics pass. You'll then need to adjust the contour to get just the right amount of 'ringing'. The result (shown in Figure 9 above) looks similar to Figure 8, but it has a distinctive quality that sets it apart from the previous patch.

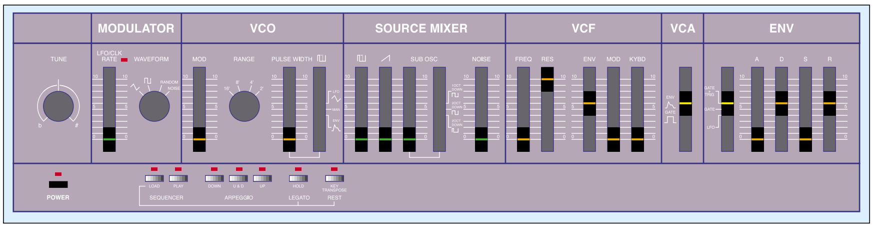


Figure 8: The SH101 kick drum.

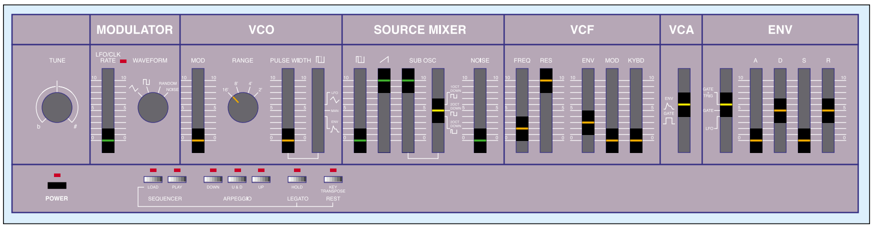


Figure 9: Another SH101 kick drum.

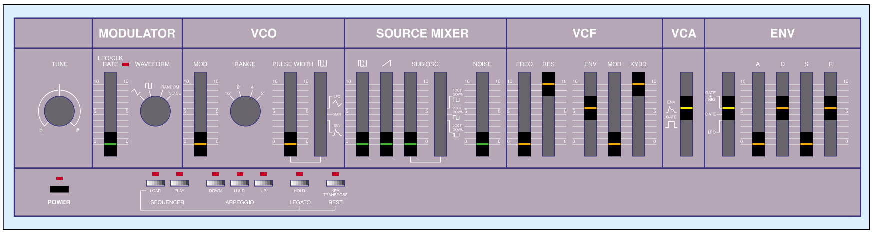


Figure 10: The SH101 SynDrum.

If you program Figure 9, you will find that it is incredibly sensitive to tiny changes in the cutoff frequency, and moderately sensitive to changes of the Envelope amount and ADSR settings. This means that there are numerous variations on the theme, allowing you to create anything from huge, booming drums to tight, snappy ones. So, before leaving the SH101 behind, I'm going to take you back to Figure 8, and make just one change, increasing the 'VCF Kybd' (keyboard tracking) fader from zero to 100 percent (see Figure 10). If you do this, and play the notes at the top of the keyboard, you'll find that you have recreated the SynDrum, an early analogue percussion instrument dating from the late '70s. This makes the characteristic 'ray gun' sound that littered the electronic music of the era. It's perhaps the cheesiest percussion sound imaginable, and you'll love it!

## How Close To Theory?

At this point, we should be able to look back to Figures 1(a) and 1(b) and see that the theory and our patches are broadly in line with one another. After all, if the Axxe and the SH101 were producing excellent bass drum sounds without a nod in the direction of last month's analysis, the academic approach would be a complete waste of time and rainforests. And it isn't.

To perform this comparison, I'll remove the trigger from Figure 1(b), and then add some annotation to remind us which bit is doing what (see Figure 11). We should now be able to relate the theory to the settings that we have chosen.

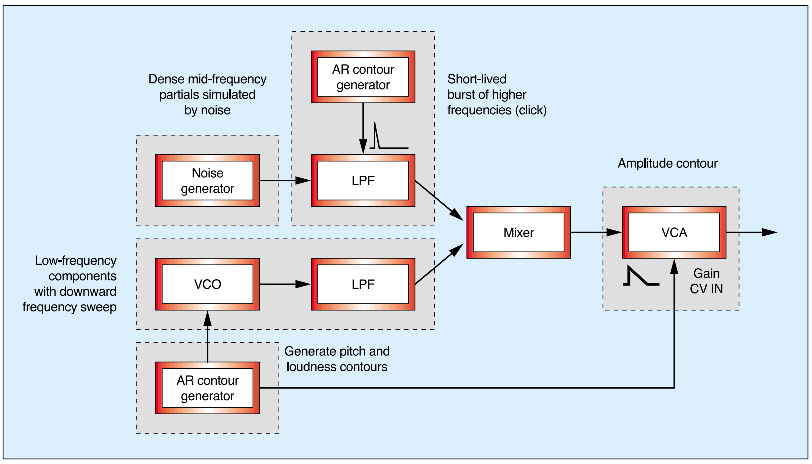

Dense mid-frequency partials simulated by noise
AR contour generator
Short-lived burst of higher frequencies (click)
Noise generator
LPF
Amplitude contour
Mixer
VCA
Low-frequency components with downward frequency sweep
VCO
LPF
Gain CV IN
Generate pitch and loudness contours
AR contour generator

Figure 11: Analysing the components in Figure 1(b).

Let's get rid of the easy stuff first. In each synth, the ADSR is providing the AR contour, and the audio VCA is responding to this as required. But what of the 'low-frequency components with downward frequency sweep'? This is the clever bit, where we replace the VCO and filter combination with the self-oscillating filter.

Next, we come to the noise module simulating the mid-frequency partials. We use the synths' noise generators for this but, as already noted, little of this sound passes to the output because of the action of the filter. Fortunately, this doesn't seem to matter, implying that the absence of the mid- and upper- frequencies does not impair our perception of the bass drum sound.

This may seem to be in marked contradiction to my statements about the importance of the mid-range partials in the bass drum sound in last month's part of this series, but I suspect that there are valid reasons for this. Firstly, audio engineers now tend to emphasise the low frequencies of the bass drum sound — whether the instrument is acoustic or synthesized. Secondly, I don't trust analogue synths, and I suspect that there are numerous low-amplitude, higher-frequency components present in the final sound (sure, we didn't ask for them, but they are introduced by distortion in the overdriven, oscillating filter and the VCA). Thirdly, we are so accustomed to electronic drums that we no longer fully appreciate the sound of a real bass drum. You can verify this easily. Load a high-quality concert bass drum sample and compare this to your synthesized sound. Now you can hear where the real mid and upper frequencies have gone.

Roland's TR909 drum machine.

Finally, there's the high-frequency click, and again we benefit from a sonic short cut. Because the attack of the ADSR is very rapid when the Attack is set to 0, the VCA creates a discontinuity at the start of the sound. For reasons we need not worry about, this discontinuity contains (or more properly *is*) a very short burst of high-frequency noise, and it's this that produces a click when the sound is triggered. I'm forever reading complaints on various on-line forums from players who bemoan the click at the start of their sounds, and write to ask whether their synths are faulty. Likewise, I've lost count of the number of times that I've answered this, explaining that it is the desirable consequence of truly snappy contour generators and VCAs.

Anyway, we can now redraw Figure 11 to show the manner in which the Axxe and SH101 produce the bass drum sound, creating Figure 12 (shown on the next page) in the process. This may look quite different from Figure 11, but the principles of the sound are identical, and it is this that allows us to develop patches for synthesizers more limited than Figures 1(a) and 1(b) would otherwise require.

## Classic Bass Drum Sounds 1: The Roland TR909

No discussion of analogue bass-drum sounds could possibly be complete without studying the way in which Roland created the most used (and over-used) drum sounds of all time. These are, of

▶

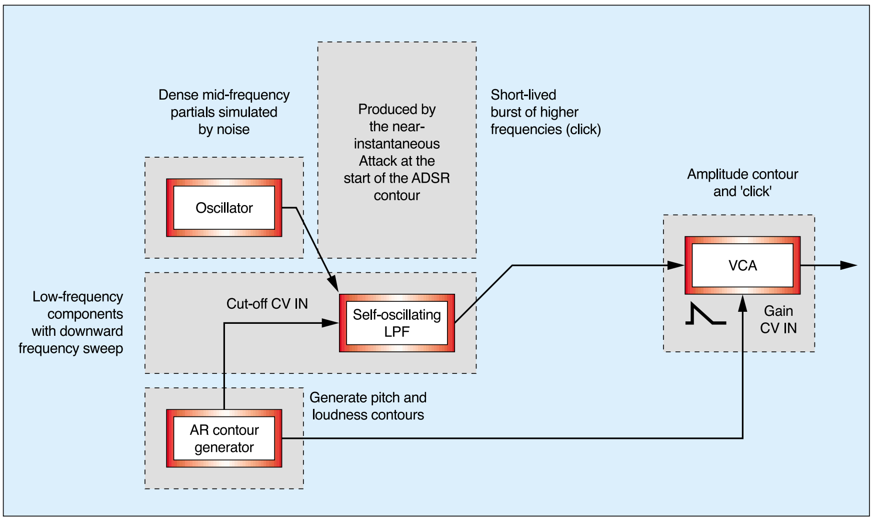
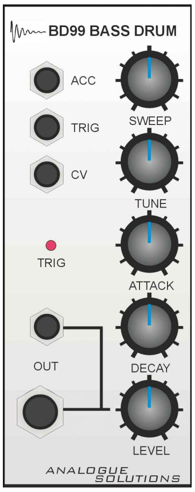

Dense mid-frequency partials simulated by noise
Produced by the near-instantaneous Attack at the start of the ADSR contour
Short-lived burst of higher frequencies (click)
Oscillator
Amplitude contour and 'click'
VCA
Low-frequency components with downward frequency sweep
Cut-off CV IN
Self-oscillating LPF
Gain CV IN
AR contour generator
Generate pitch and loudness contours

Figure 12: How the Axxe and SH101 recreate the sound of Figure 1(b).

▶ course, the bass drums generated by the TR808 and TR909.

We'll start with the TR909, because this is the one that most closely follows the principles that we have discussed (see Figure 13 — and before anybody writes in to say that this doesn't look like Roland's schematic... you're right, I've laid it out differently).



Analogue Solutions' BD99 909 kick drum synth module.

Starting with the upper signal path, you can see that the oscillator produces a sawtooth wave whose pitch is defined by EG3, which has an instant Attack and slow Decay. The output from the oscillator then passes through a waveshaper. This removes almost all the overtones, transforming the sawtooth into something very close to a sine wave. This in turn passes through a VCA controlled by another contour generator (EG1) that provides the required AR envelope (the amplitude of EG1's Attack and its Decay rate are modified if the Accent voltage augments the Trigger).

So far, so good... so let's now turn our attention to the lower audio path. This starts with a noise generator whose output passes through a low-pass filter to remove the high frequencies. The output from this is mixed with a short pulse (essentially a click) provided by the pulse generator, and the sum of the two is contoured by a VCA controlled by EG2. Finally, a mixer combines the output from both the upper and lower signal paths to create the composite sound.

It may take a couple of moments to digest Figure 13, but once you have done so, it should be clear that Figure 1(b) and Figure 13 describe remarkably similar systems. Sure, there are detailed differences, but the fundamental ideas and patch structure are common to both.

Some synth fanatics dig deeply into the electronics of their TR909s to add additional controls for each element of the patch in Figure 13. However, if you don't fancy doing this, you could buy an Analogue Solutions BD99 (shown below left), a module that duplicates Roland's original circuitry, but provides a number of additional facilities.

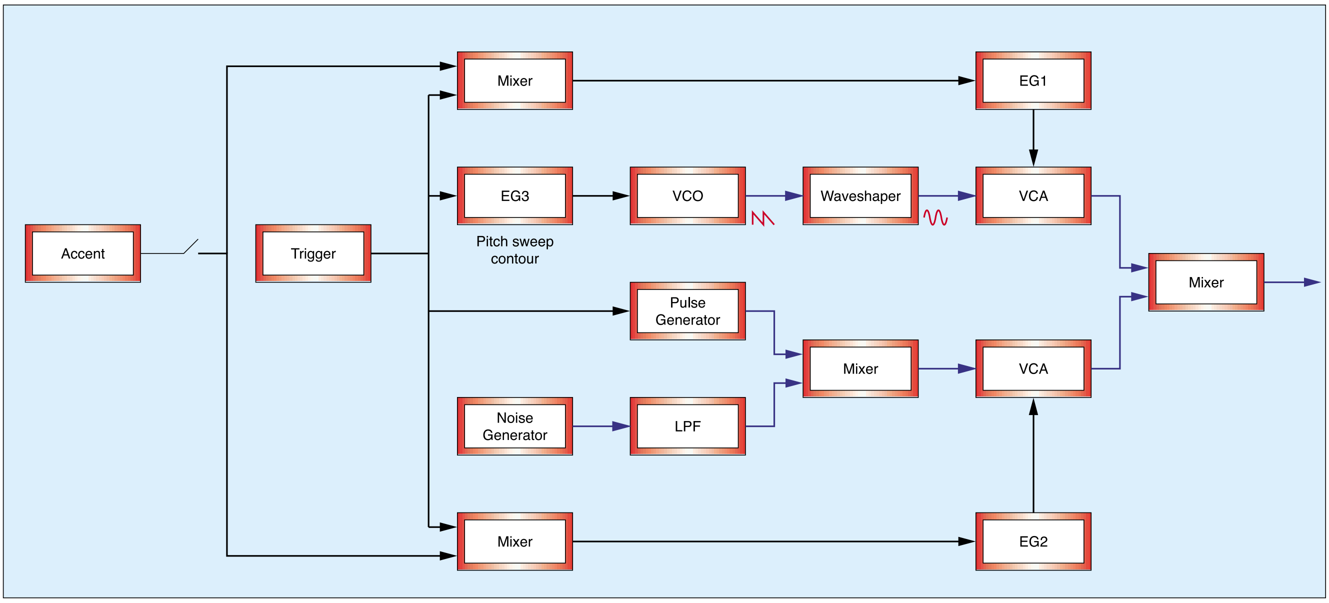
For example, you can tune the basic pitch of the VCO (something that was not possible on the TR909). You can also control the amount of 'click' heard at the start of the note (the Attack level of EG2) and the Decay time for the sound (the Decay rates of EG1 and EG2). You can even apply varying amounts of Accent using the input provided, and control the pitch of the VCO using a CV.

Of course, none of this alters the basic principles we have discussed... these controls simply allow you to change some of the parameters that define the exact nature of the sound. But what a difference this can make. Many percussion instruments share common principles, so you can leave the domain of bass drums far behind, and use the BD99 to produce sounds such as toms and congas too.

## Classic Kick Drum Sounds 2: The Roland TR808

Given that many people lump the TR808 and TR909 together, you may be surprised to discover that they generate their sounds in

Figure 13: The Roland TR909 bass drum.



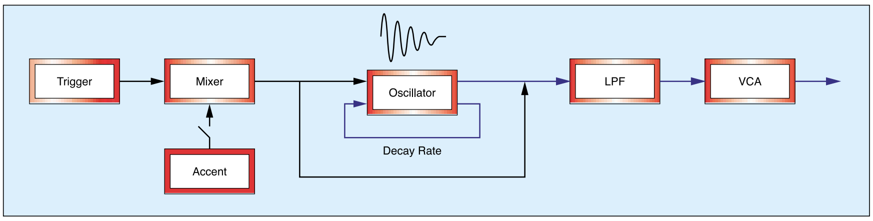


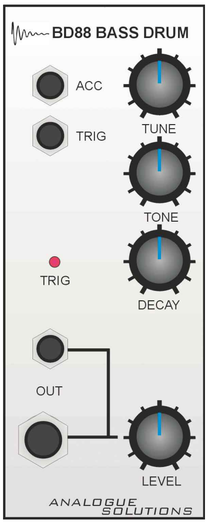
Figure 14: The Roland TR808 bass drum.

The (inter)face that launched a billion kicks: the TR808.

entirely different ways. Indeed, whereas the TR909 kick drum is virtually a synthesizer in its own right, the TR808's is a far simpler — and in many ways far cleverer — bit of electronics.

Figure 14 (above) shows the block diagram for the TR808 bass drum and its slightly more advanced Analogue Solutions progeny, the BD88. As you can see, it lacks the multiple signal paths we have discussed, has no noise generator, and no contour generators. So how does it produce the sound we want? To answer this, we must look at the type of oscillator used in the circuit. It is called a 'Bridged T-network', and it is quite unlike any oscillator that we have encountered before in Synth Secrets.

Consider the oscillators in your analogue synthesizers, whether monophonic, polyphonic, paraphonic... or whatever. All these synths' oscillators produce their outputs continuously, and whether we hear them or not is determined by the action of VCAs controlled by contour generators which are themselves initiated by triggers, as shown in Figure 15 below (if you connect the output from a conventional oscillator directly into a PA system, it will howl at you unceasingly until you pull out the plug).



Analogue Solutions' BD88 TR808 kick drum module.

Now cast your minds back to junior school. Did you annoy your teacher by holding your ruler hard against the desktop before flicking it to go "boooooiiiiiigggggg"? If you did, you were using an oscillator that, once excited, produced a sound that decayed to silence, yet did so without *any* contour generators or amplifiers.

It should come as no surprise to learn that some electronic circuits respond in the same way. The Bridged T-Network in the TR808 and BD88 is one such circuit, so we can replace Figure 15 with the simpler diagram that is Figure 16 (below right).

With an oscillator of this sort, you can use positive feedback to determine the decay characteristic of the sound. If you increase the amplitude of the feedback too far, the decay will extend to infinity, and we will be back where we started, with a continuous oscillator.

Understanding this, we should now be able to decipher Figure 14. The Trigger kicks the oscillator into life, initiating the audio signal (the presence of an Accent increases the impulse and, through the action of some very clever circuitry that we will not discuss here, also makes the sound punchier by accentuating the start of the note). Next, some of the Trigger+Accent signal — which is, of course, a brief CV pulse — is added into the audio signal path to emulate the beater hitting the membrane of the drum. The combined signal then passes through a low-pass filter that allows you to remove higher frequencies, thus subduing the amount of click if desired. The final VCA then amplifies the signal and feeds it to the output.

Finally, it's worth noting that, by accident or design (I'm not sure which), the TR808 kick drum oscillator goes slightly flat at long decays, which is exactly what's required to make the patch sound convincing. Earlier Roland rhythm products (such as the CR68 and CR78) did not do so, and this is one reason why analogue drum machines sounded so much better from the TR808 onwards.

## How Low Can You Go?

So there we are... the secrets of analogue bass drums laid bare. And, although the TR808 bass drum circuit shows that there is more than one way to skin this particular cat, we've never strayed too far from the theories laid down last month. All of which goes to show that, if you understand the nature and fundamental characteristics of a sound, you can take a good shot at recreating it on any synth. Moreover, as Roland discovered with the TR808 and 909, you may even develop a sound that will become a sonic masterpiece in its own right. Now, wouldn't that make it all worthwhile?

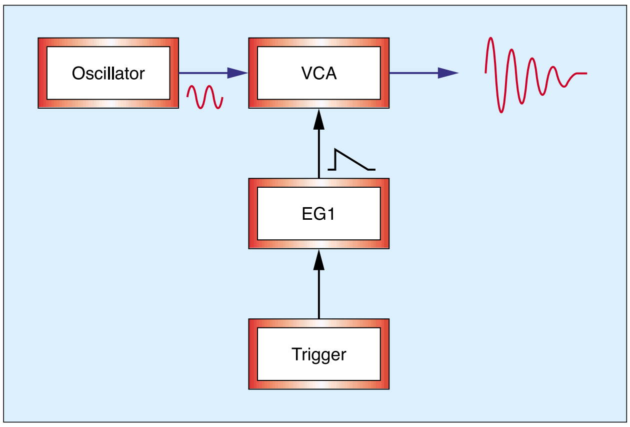


Figure 15: The conventional signal path: trigger, envelope, oscillator, and amplifier.

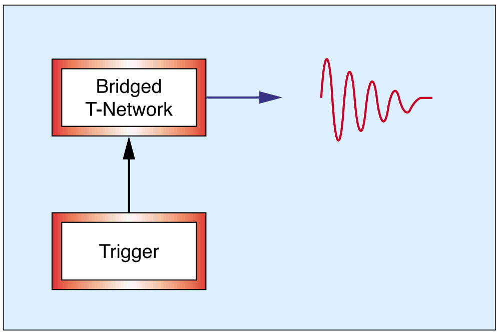
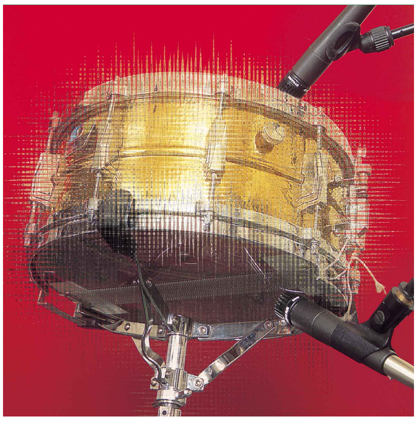


Figure 16: An oscillator that decays to silence on its own.

# Synth Secrets

**If you thought synthesizing realistic bass drums was complex, that's *nothing* compared to snares. So how is it that the analogue snare sound is so well known? And how *do* you go about creating it? We find out...**



## Synthesizing Drums: The Snare Drum

Gordon Reid

For the past two months, we've been discussing the science and synthesis of bass drums, so it is almost inevitable that we're now going to move onto the snare drum. Actually, strike the word 'almost' from the last sentence. It *is* inevitable. After all, these two types of drum form the basis of almost all the percussion tracks in pop and rock music. Sure, we need metalwork such as hi-hats and cymbals to embellish the rhythm, and toms provide needed variation for fills and other effects, but it is the bass and snare combination that provides the drive and 'oomph' of most percussion tracks.

Before embarking on any discussion of the snare drum, I recommend that you go back over the instalments of this series from last November, and January this year (see www.sound-on-sound.com/sos/nov01/articles/synthsecrets1101.asp and www.sound-on-sound.com/sos/jan02/articles/synthsecrets0102.asp respectively). This is because many of the concepts used this month were explained in detail in those articles, particularly those concerning quasi-harmonic series and the use of the frequency-shifter.

Now, armed with an understanding of pitched membranophones and — more importantly — an unpitched membranophone (the bass drum) we're ready to begin our analysis of the snare drum.

Like the orchestral bass drum, the snare has two complete heads. And, as on the bass drum, the upper surface is the *batter* head. However, whereas the bass drum had a carry head, the lower head on the snare drum is called the *snare* head. This is because, unlike a carry head, it has a snare of chains or cables stretched across it, and it is these that give the drum its distinctive sound. Indeed, without the snares, the snare drum sounds much like a shallow, dual-headed tom — and in fact, we'll start this month by omitting the snare and considering the drum as if it *were* a tom.

### The 'Snare-Less' Drum Modes

If you've just re-read January's instalment of this series, or have been following my previous analyses of drums month by month, you will not be surprised if I tell you that the snare drum's two heads are coupled by the enclosed air between them. Furthermore (and this is not something that we have discussed in the past) the heads are also coupled by the shell of the drum itself. This means that the frequency distribution of the drum's modes is quite unlike the modified Bessel distribution of a single suspended membrane.

Experiments show that, like the orchestral bass drum, the snare drum produces two frequencies for each of the 0,1 and 1,1 modes, and that the modes are distributed enharmonically. To illustrate this, I have shown in Figure 1 (right) the nine frequencies produced by the first seven modes of a typical 'snare-less' snare drum.

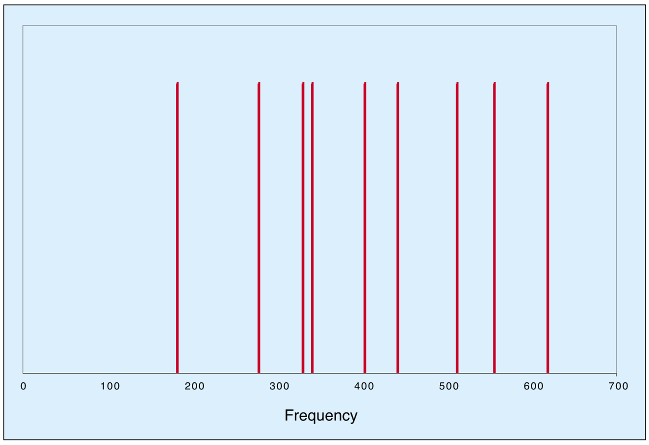
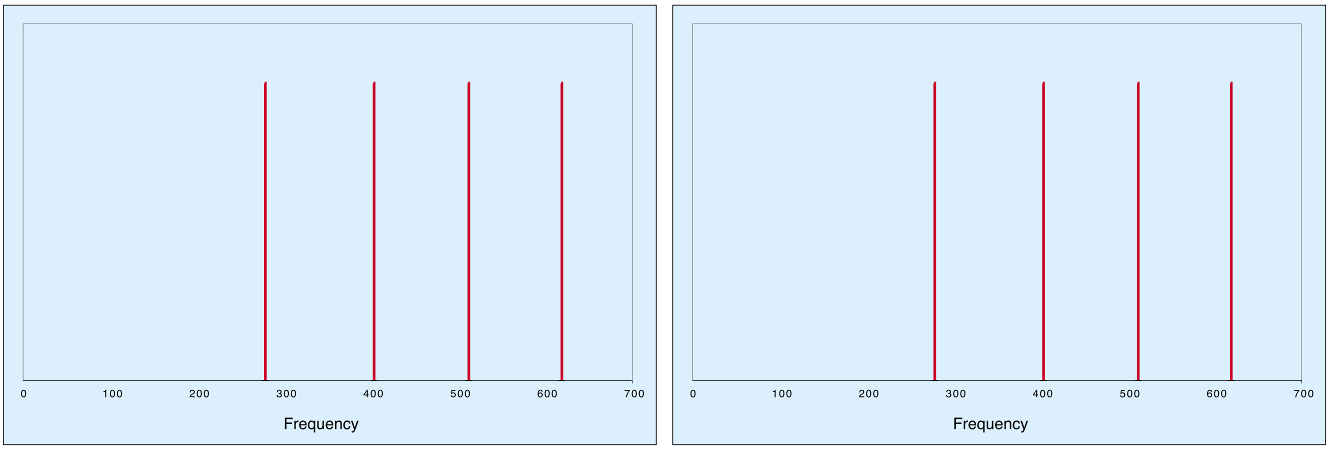


Figure 1: The nine frequencies produced by the first seven modes of a snare drum.

As you can see, the modes have no obvious relationship to one another. However, if I ignore the two components at approximately 180Hz and 330Hz produced by the 0,1 mode (I'll come back to these in a moment), and separate the remaining frequencies into two series, something interesting happens (see Figures 2(a) and 2(b), right).

As you can see, the result is two series whose partials appear to be evenly spaced. This should, perhaps, be no surprise. When we did the same for the bass drum two months ago, we obtained a similar quasi-harmonic series. OK, the snare drum appears to produce two such series, rather than the bass drum's single series, but the principle remains the same.

Unfortunately, the near-harmonicity of the snare drum's two series is an illusion, because the distances between the components in Figure 2(a) are 125Hz, 109Hz



Figures 2(a) and 2(b): The shifted quasi-harmonic series produced by the snare drum shown in Figure 1.

and 107Hz, and those in Figure 2(b) are 101Hz and 114Hz. Nonetheless, the distribution is even enough to suggest a way to use harmonic oscillators to imitate the 'non-snare' part of the snare drum sound.

Figure 3 (below) shows the method I used to create the quasi-harmonic series for the bass drum in January, and it is equally applicable to (or indeed equally inapplicable to) each of the two series in Figures 2(a) and 2(b). The difference is that we need two frequency-shifters producing different amounts of shift; one for each of the series.

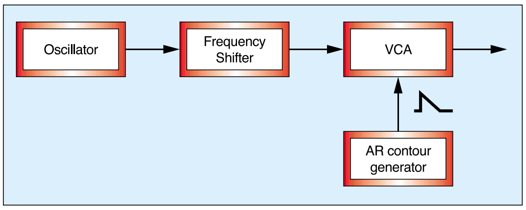


Figure 3: Generating a 'shifted' harmonic series.

If we also append a couple of sine-wave oscillators to add back the 0,1 modes that I rather casually removed a few paragraphs ago, we obtain the architecture shown in Figure 4 below, and the spectrum shown in Figure 5(a) on the next page.

At this point, you may wonder how I obtained the oscillator frequency of 111Hz in Figure 5(a), and the frequency offsets of 175Hz and 224Hz. There's nothing mystical about this; I just calculated the closest fit for the frequency differences between the components in the series (which turns out to be 111Hz) and the offsets that provide the best matches to the frequency table from which Figure 1 was drawn.

We can now see how closely (or not) this conforms to the true spectrum — see Figure 5(b). You will probably agree that — on paper, at least — my synthesis corresponds reasonably well to reality, although I have no doubt that your ears would be able to tell the difference.

Unfortunately, the potential modular synth in Figure 4 is rather complex. Furthermore, due to the presence of the dual frequency-shifters, it's more than a little pricey.

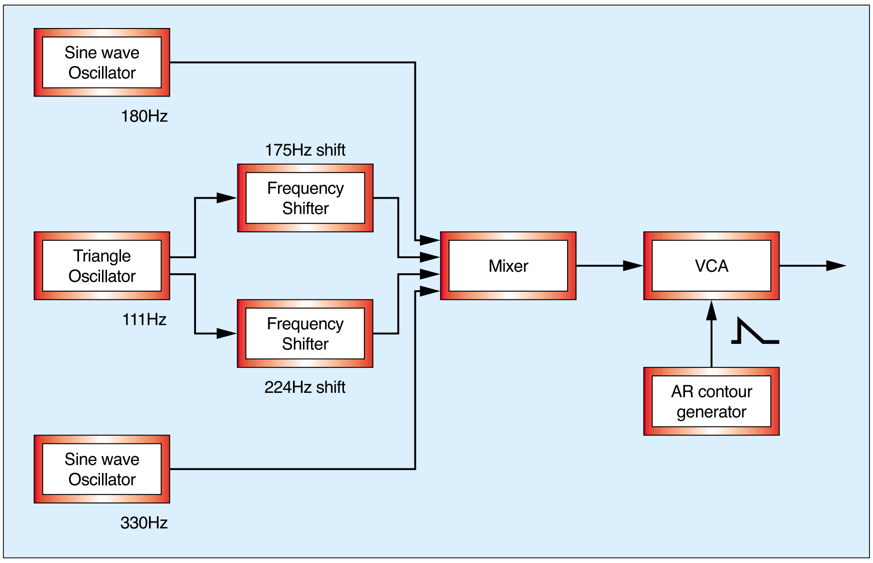


Figure 4: Generating two shifted series plus the rogue 0,1 modes.

Unfortunately — again — if we stick to this method of synthesis, there is little or nothing we can do to reduce the cost. Indeed, measurements of real snare drums suggest that, for a realistic sound, we must add further modules and *increase* the complexity rather than reduce it. This is because the frequencies produced by the 0,1 mode decay far more quickly than the other partials... sometimes at more than twice the rate. Although this means that they are rather short-lived (which makes it tempting to eliminate them altogether), we can't do this, because it would remove depth and bottom end from the sound. So, to synthesize the correct response, we have to pass the outputs from the sine-wave oscillators through a second VCA whose gain is controlled by another Attack-Release generator, one with a quicker Release than the first one (see Figure 6 on the next page).

At this point, the architecture in Figure 6 is starting to look a little too elaborate for my tastes. OK, it's not the most convoluted patch I've ever created for Synth Secrets, but given that it's just a single element within a more complex sound, I'm worried that things might get out of control.

So let's look at this sound in a different way. After all, we are considering a limited number of partials, and that suggests that additive synthesis might provide a more suitable solution to the problem. But even the nine partials in Figure 1 require nine oscillators, nine amplifiers and nine contour generators. These would allow us to program exactly the right frequencies, amplitudes and decay rates, but, including the final mixer, this would require 28 modules (see Figure 7). Given the high price of frequency-shifters, this may still work out somewhat cheaper than the 12 modules shown in Figure 6, but it's far from a practical solution. Oh yes...

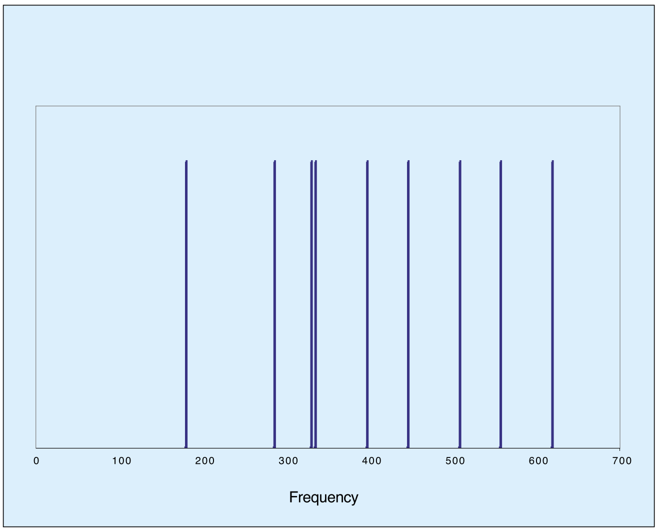


Figure 5(a): The snare partial series generated by the synthesizer in Figure 4.

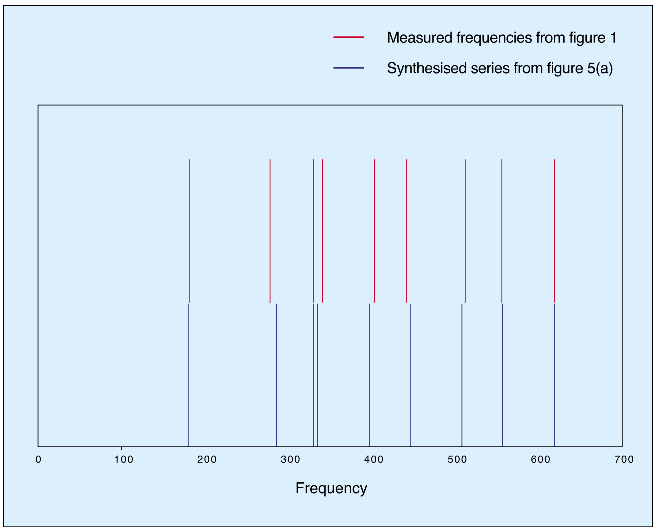


Figure 5(b): Comparing the true and synthesized series.

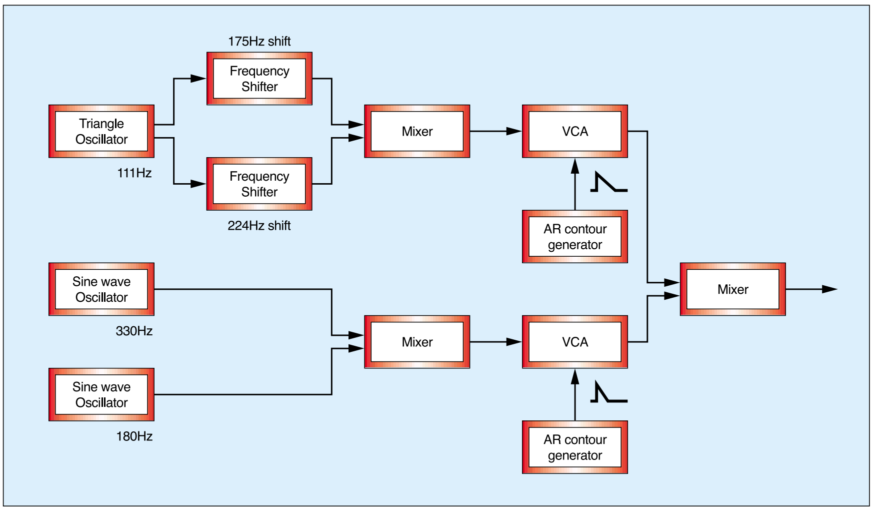


Figure 6: A better snare-less snare drum synthesizer.

and my choice of working with nine partials is arbitrary. In reality, the drum produces scores of partials. So it seems that we are stuck with Figure 6.

Unfortunately, no integrated synthesizer can create this patch, suggesting that the *only* way to program snare drum sounds is to use racks full of modules from manufacturers such as Analogue Systems and Doepfer, or spend ages messing around with flexible software synths such as Native Instruments' *Reaktor*. Yet experience tells us that we can create reasonable approximations to snare drums using fairly basic synthesizers. So maybe there's another way?

Two other mechanisms that suggest themselves are FM synthesis and ring modulation. We know from previous discussions that both of these methods will produce large numbers of enharmonic partials, and these may be suitable for synthesizing the semblance, if not the reality, of the drum's modes. Unfortunately, I can find no combination of Carrier and Modulator that produces the correct distribution of frequencies. So, for the moment, there doesn't seem to be much point pursuing these methods further, and we appear again to have reached a dead end.

## The Effect Of The Snare

So far this month, we've done little to move beyond the concepts of drums expounded in previous Synth Secrets. We've simply applied old ideas to a new drum of different size and proportions, but one which exhibits very similar physics to those described before. But now we're going to consider the snare itself — the arrangement of cables stretched under the drum.

Let's first study the properties of a stretched circular membrane that has a single cable of some sort stretched just underneath its lower surface, as shown in Figures 8(a) and 8(b).

Let's now consider what happens to the centre of the drumhead, and to the centre point of the cable, if we displace them by, say, smacking the upper one with a large stick. Since we're striking the drumhead directly from above, we force it down, distorting it from its rest position. At some

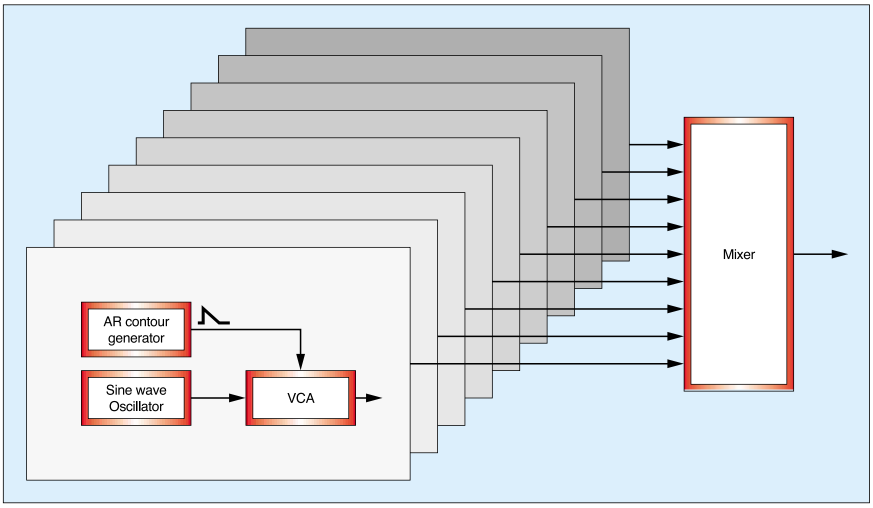


Figure 7: The additive approach to generating drum partials.

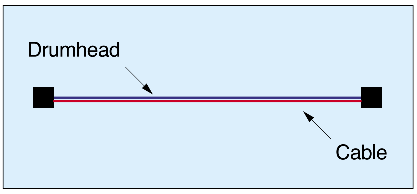


Figure 8(a): A drumhead and simple snare at rest.

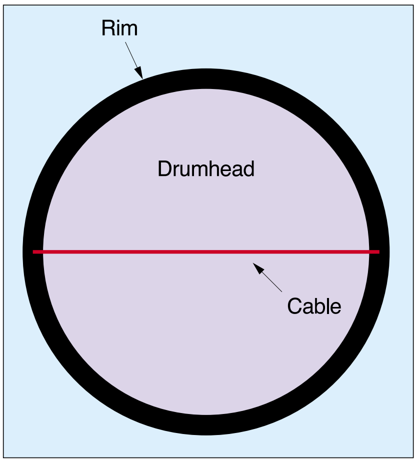


Figure 8(b): A drumhead and simple snare at rest (top view).

point in its downward travel, it will then strike the cable, and push this from its rest position. Provided that the drumstick is still pushing down on the head, the two will then continue to distort together until the stick bounces off.

Now, since both objects are under tension and (to a greater or lesser extent) elastic, both will try to return to rest. Of course, they don't do this instantaneously, and both will oscillate above and below their rest positions until all their energy is dissipated.

Let's now assume that, by some coincidence of mass, elasticity and tension, the head and the cable respond in similar fashion, oscillating as shown in Figure 9 (right). This shows that the head starts to return to rest as soon as the stick leaves its surface, while the cable continues to move downward for a short time longer before it too oscillates and returns to rest.

In this scenario, the head and the cable do not interact with each other after the initial impact and displacement. This, as you will appreciate, is not very interesting. It simply describes two dissimilar oscillators that happen to produce the same frequencies as they radiate away their energy.

However, since drumheads and cables are very different objects, this scenario is extremely unlikely. Indeed, since the head is a two-dimensional oscillator whose oscillations are described by modified Bessel functions, and the cable is a one-dimensional harmonic oscillator, I would venture (without any proof, you understand) that it is impossible. So we now have to consider what happens in the real world when the head and the cable attempt to oscillate at different frequencies.

Sometime after the initial stick impact, the cable will leave the lower surface of the head and then, some time later, snap back to strike it. At this moment, the head and the cable will pass energy to each other, exciting new modes of vibration, and thus new oscillation frequencies. A short time later, they will strike each other again, exciting yet more new frequencies, and so it goes on until all the energy is dissipated. I have shown a hugely simplified representation of this in Figure 10 below.

If the oscillations of the head and cable in Figure 9 are uninteresting, those in Figure 10 are anything but, so much so that it's impossible to analyse what these might be. Indeed, a tiny difference in the tension of either object or the initial velocity of the stick can radically change the timing and nature of the impacts that occur as time progresses.

Now, it's a small jump of intuition to realise that the drumhead and the cable are adjusted optimally when the maximum number of head-on collisions occurs between them. This type of impact (which happens when the head and snare are moving in opposite directions) ensures that the greatest number of high-frequency modes is generated, and that the energy is radiated away most quickly. Both of these factors are vital in producing the characteristic 'snap' of the modern snare drum.

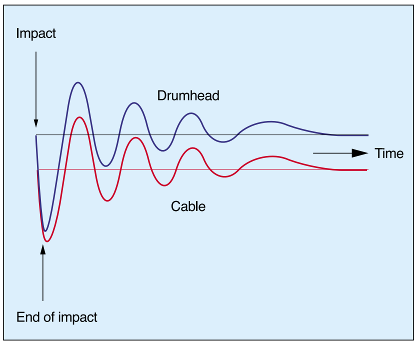


Figure 9: The head and cable do not interact.

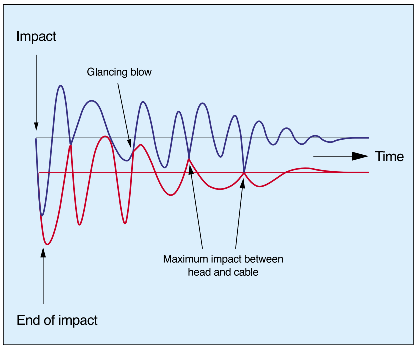


Figure 10: A hugely simplified representation of the interaction between a drumhead and a cable.

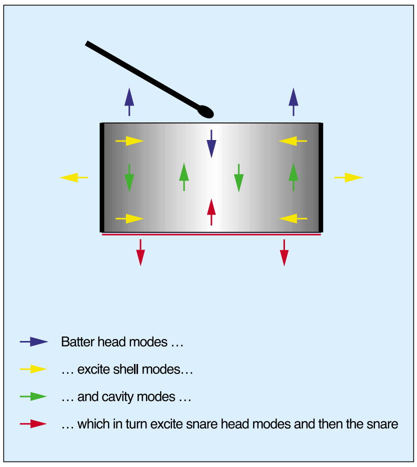

Batter head modes ...
... excite shell modes...
... and cavity modes ...
... which in turn excite snare head modes and then the snare

Figure 11: The way in which the snare is excited.

Just to demonstrate how complex a real snare drum is, I should now remind you that up until now, we have been considering *only* the centre point of the snare head and the centre point of the cable. So now you must try to envisage how complex the motion becomes when different parts of the head are moving in different directions, different parts of the cable are moving in different directions, and numerous impacts are occurring all along the length of the cable. The resulting motion defies analysis, as does an analytical derivation of the modes and frequencies thus excited. Indeed, if an impact occurs at one point along the cable, it momentarily changes the length of the cable, so *any* frequencies above the fundamental (subject to a high-frequency limit determined by factors such as mass and elasticity) are possible. This means that the cable is *not* acting as a harmonic oscillator; it is closer to some form of band-limited noise generator.

Now let's make things even more complex. When playing a snare drum, we do not hit the snare head directly, and it is up to the shell and the air cavity to excite the snare head after we strike the batter head. This means that different parts of the snare head will move in different directions as soon it starts oscillating. I have shown a hugely simplified representation of this in Figure 11.

Of course, nothing is ever this simple (ha! ha!) and, once the batter head has excited the shell and the cavity, these in turn excite each other and the batter (again) as well as the snare head and the snare cable. And, inevitably, each time the cable hits the snare head, it excites new modes in the shell and cavity... and so on, and so on. To say that the snare drum is a hugely complex system of resonators is a bit like saying that space is big. The words are correct, but they convey no sense of the scale of things.

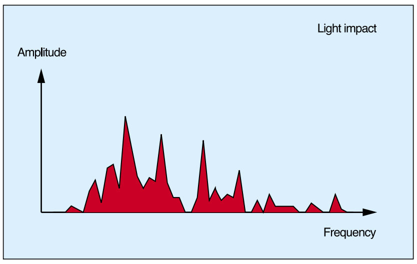


Figure 12(a): The spectrum of a lightly hit snare drum.

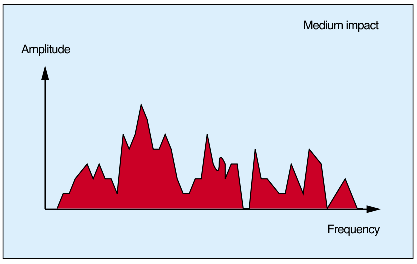


Figure 12(b): The spectrum of a snare drum hit with medium force.

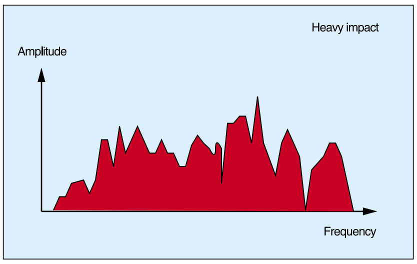


Figure 12(c): The spectrum of a heavily hit snare drum.

## Frequency Response Of The Snare Drum

Inevitably, such a complex system produces a hugely complex spectrum. Unfortunately, the sound generation mechanism is *so* complex that it defies the kind of analysis we've been performing over the past few months. Even academic text books shy away from this, so I think it's fair to say that, not only is a precise analysis of the snare drum outside the scope of Synth Secrets, it's outside the scope of, well... everyone. And if this is true for a single, defined strike of the batter head, think how more complex and impenetrable the subject becomes when you consider the different interactions of the shell modes, the snare head and the snare when you strike the batter head with a different velocity, or in a different position.

Clearly, our current (if somewhat simplified) understanding of the snare drum — while correct — is insufficient for our needs, and we need a different approach if we are to progress further. So let's ignore things we can't analyse, and consider what we *can* state about the sound of snare drums.

Fortunately, this is where the mist begins to clear, because without knowing the mechanisms involved, we can make two important generalisations with confidence. This is because the human race has a great deal of empirical knowledge about the snare drum, ie. people have been playing them for an awfully long time, we've all heard lots of them, and they all conform to these generalisations.

The first of these is that the harder you strike a snare drum, the louder it becomes, and the more energy is radiated at higher frequencies. Figures 12(a) to 12(c), which show three measured spectra of a single snare drum, demonstrate this clearly. They show how the energy in the first few hundred Hertz of the spectrum increases as the drum is struck harder, and how a greater proportion of this energy appears at higher frequencies.

You may also note from these diagrams that the spectrum becomes more noise-like at higher impact velocities. This is because a *very* lightly hit snare drum will barely excite the snare, so the drum modes dominate its sound. The light impact shown in Figure 12(a) will excite the snare, but the drum modes still dominate the sound. As the drum is struck harder, the widths of the peaks widen, and the sound becomes more 'noisy' as the cable's interaction with the drum becomes significant. Finally, when you strike the snare drum very hard, as shown in 12(c), the bulk of the energy is contained in a wide, noisy spectrum. So here is our second generalisation: the more that the snare interacts with the drum, the wider the modes become, and the 'noisier' the sound becomes, eventually changing into a complex noise spectrum. I suspect that most people have noticed this without ever having quantified the differences they're hearing.

Amazingly, it is these simple observations and measurements — not our complex physical analyses — that tell us how to synthesize the snare drum's sound. Given that we will usually want to emulate the high energy sound of the heavy impact (you want your drummer to smack the things with gusto, don't you?) we need to use a noise generator!

The simplest such solution would be to pass the noise though a simple VCA and EG combination, as shown in Figure 13. However, while this might seen elegant and simple, it is inadequate for accurate snare drum emulations, so we need to extend the architecture somewhat further.

Figures 12(b) and 12(c) show how the amount of high-frequency energy increases the greater the impact. We can simulate this by placing a low-pass filter in the signal path, and raising and lowering the cutoff frequency according to the impact velocity we wish to simulate. It would also be interesting to place a few band-reject filters (notch filters) in the architecture (Figure 14) in order to synthesize the holes observed in the snare drum spectrum. I have placed these under voltage control so that we can move the holes around if we wish to.

This is almost certainly sufficient to produce a snare-like sound, but if we want to be sophisticated, we can put in the sound-generating mechanism from Figure 6 to that in Figure 14. We can then add a velocity-controlled mixer, so that the proportions of the snare-less drum modes and snare-affected sounds are balanced dynamically. We now have a reasonably sophisticated model of the snare drum (see Figure 15, above right).

Of course, this is not the end of the story, because there are many other factors that influence the sound produced by the drum. For example, we have considered the snare to be just a single cable running underneath the snare head. Modern snares comprise many parallel cables or chains, and there is no reason why these should be stretched to the same tension. Even if they are, they will not lie in exactly the same place, so they will be in different positions with respect to the snare head's modes, and will therefore hit the snare head in different positions and at different times.

Furthermore, changing the tension of the snares can have a dramatic effect, creating very different energy distributions. Indeed, if

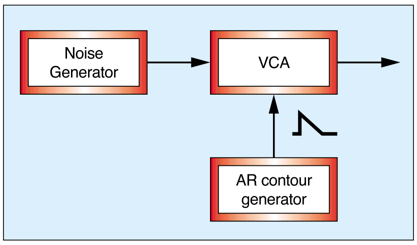


Figure 13: A noise-based snare drum sound.

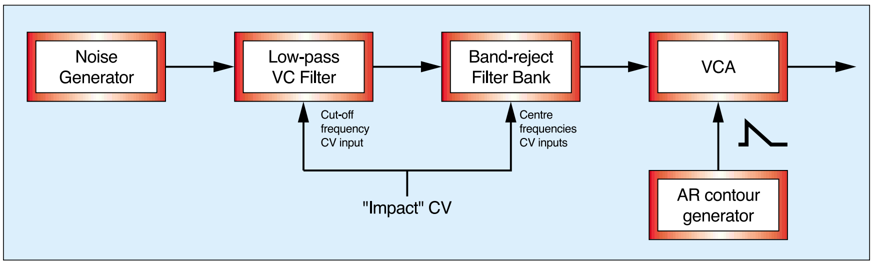


Figure 14: Improving the snare sound in Figure 13.

the snare is too tight, it will be almost impossible for it to part company from the snare head, and there will be no subsequent impacts to create the characteristic snare drum sound. On the other hand, if the snare is too loose, it will flap around aimlessly, and the sound will again be changed for the worse.

Then there are factors such as the size and depth of the drum shell, the tensions of the batter and snare heads, the materials from which the shell is constructed... and many, many others. In fact, even the way in which the drum is supported will have a noticeable effect, and a snare drum on a rigid stand will sound different from the same drum supported on a flexible, elastic mount. This is because, on the conventional stand, a significant amount of energy is lost through mechanical coupling of the drum to the stand itself.

So now it's time to move on, firm in the knowledge that, while we understand very little of the detailed interactions that occur in a snare drum, we are in a position to synthesize a reasonable approximation of the sound. But I'm afraid that that will have to wait until next month...

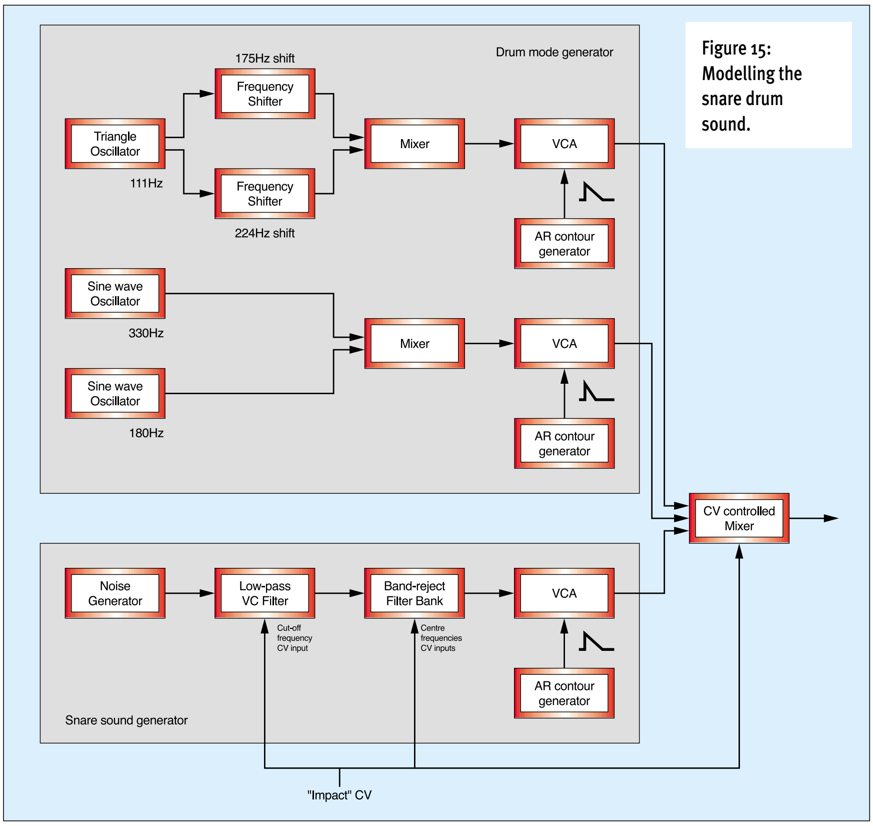
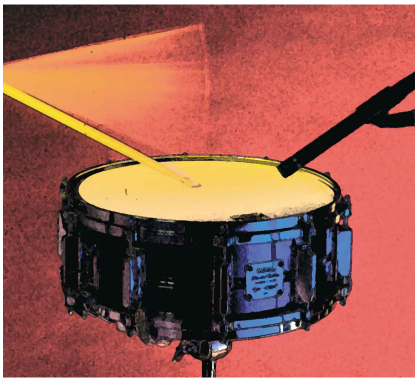


**Figure 15: Modelling the snare drum sound.**

# Synth Secrets



**Last month, we revealed just how hideously complex the sound-producing mechanism of the snare drum can be. Nevertheless, synthesizing the sound is not as hard as it seems, as we find out with the aid of a Roland SH101...**

## Practical Snare Drum Synthesis

Gordon Reid

We finished last month's Synth Secrets with the diagram shown below, which depicts one possible model for using analogue synth modules to recreate the sound of a snare drum. This model includes five primary sound sources, numerous contour generators and VCAs, some filters, a couple of frequency-shifters, some preset mixers, and a voltage-controlled mixer that allows you to alter the contributions from each signal path.

If you decided to build this patch using an analogue modular synth, you'd have to spend about £2000 on racks and modules, but it's well known that you can generate acceptable 'analogue' drum sounds from small monosynths costing a fraction of this. Indeed, you can pick up a Roland TR808 or TR909 for a few hundred pounds, and each of these will give you a drum sequencer, bass drums, toms and all sorts of metalwork and percussion in addition to their snare drum sounds. So how do these machines achieve this?

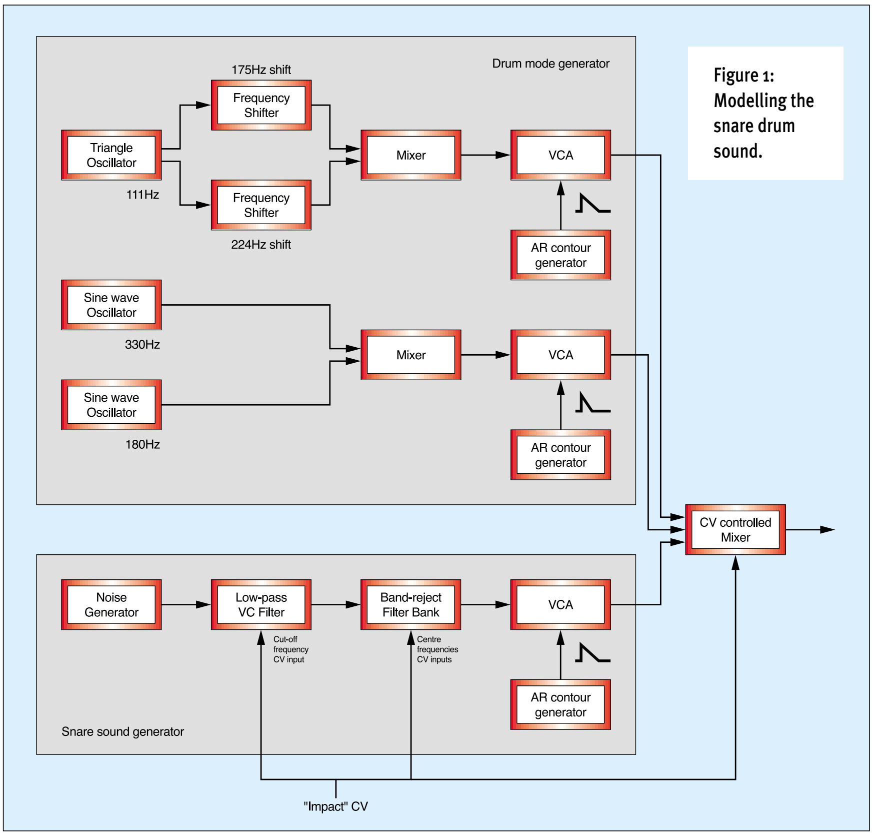


**Figure 1: Modelling the snare drum sound.**

### The TR909 Snare

Let's start by looking at the snare drum in the Roland TR909 (see Figure 2, right). This doesn't look much like Figure 1, but let's break it down into its components and see how close it is. We'll start with the subset of 11 blocks to the centre and right of the diagram, as shown in Figure 3.

To understand these more easily, we need to redraw Figure 3 in the format I've been using in Synth Secrets for the past three years. The result is Figure 4, also shown right.

Let's now compare this to the upper signal path in Figure 1. You may remember that the partials generated by the shell and air modes of the snare drum fell into two camps: two shifted harmonic series, plus a pair of enharmonic partials generated by the 0,1 mode. I devised two signal paths to recreate these: a triangle oscillator and two frequency-shifters to generate the shifted series, and two sine-wave oscillators to generate the 0,1 frequencies.

You may also remember that I admitted that the frequency-shifters in the upper path were an expensive solution to the problem of recreating the snare sound. Well, Roland's engineers found a cost-effective solution when they designed the TR909: they ignored the shifted series, and just employed two oscillators and two waveshapers to generate the 0,1 frequencies. So there's our first compromise... we've dispensed with the elements of the sound that occur as a consequence of all the shell modes except 0,1. This may seem rather Draconian, but it turns out that it's not a bad solution, as the sound of the TR909 demonstrates. It also shows that elements of a sound that may seem vital in an academic analysis can

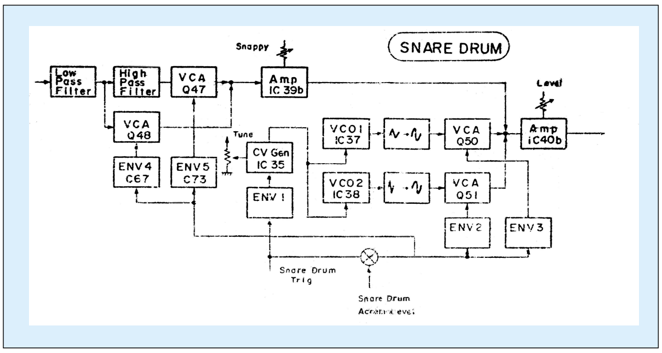


Figure 2: Roland's block diagram for the TR909 snare drum.

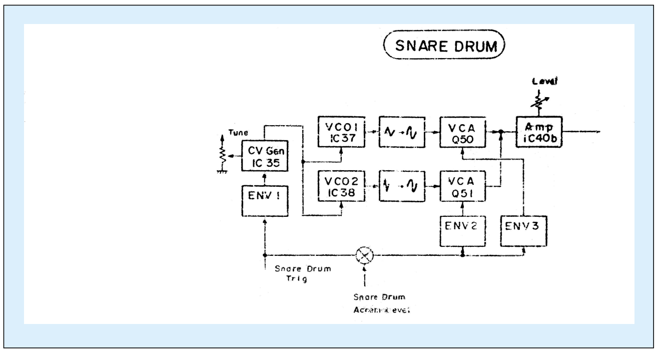


Figure 3: A subset of the block diagram for the TR909 snare drum.

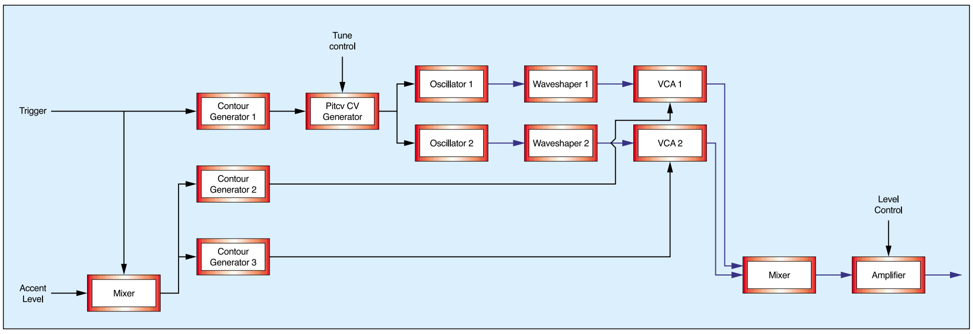


Figure 4: A different way to represent the architecture shown in Figure 3.

become dispensable when we actually begin to synthesize a sound.

Returning to Figure 4, you may have noticed that Roland added an enhancement not present in Figure 1. Whereas I used a single VCA and contour generator for the 180Hz and 330Hz partials, they provided two VCAs and contour generators. These allow the two partials to have different amplitudes and decay at different rates, as they would on a real snare drum. On the other hand, the pitch CV and associated contour generator in Roland's architecture appear to be anomalous. Given that a real snare tends not to display a pronounced decay in pitch as the amplitude of the note decays, they seem to be unnecessary.

Despite these minor differences, I hope it's clear that this part of the TR909 conforms closely to the analysis I performed last month. So let's now look at the remainder of the sound-generating mechanism. Figure 5 (below left) shows the rest of Roland's block diagram, and Figure 6 converts it into Synth Secrets format.

Here we can see that a low-pass filter modifies the output from the noise generator by removing the higher frequencies from the signal. The output from this filter then proceeds down two paths. The first takes it to a high-pass filter that removes the low frequencies from the signal, leaving a narrow band of noise that is then contoured by VCA 4 and Contour Generator 5. The second signal path travels through VCA 3, and its amplitude is shaped by Contour Generator 4 without any additional filtering. A mixer then recombines the two paths, after which another VCA further shapes the sound before it passes to the final mixer and the output.

This is quite clever, because it allows the TR909 to generate a noise spectrum with different amplitude and decay characteristics in the high- and low- frequency regions. And, while the result is not as sonically complex as the output from the filter bank in Figure 1, it nonetheless adds interest to the sound. Having accepted this difference, the rest of the 'noise' path in the TR909 snare sound is all but identical to Figure 1.

Let's now put Figures 4 and 6 together, making Figure 7 on the next page. This may not look much like Figure 2, but it is functionally identical. More to the point, it doesn't look much like the snare model in Figure 1, but closer inspection shows it to be very similar indeed. So let's take our hats off to Roland's engineers, because for a tiny fraction of the price needed to create my patch, they created a snare sound that was cheap, conformed to theory, and sounded good too.

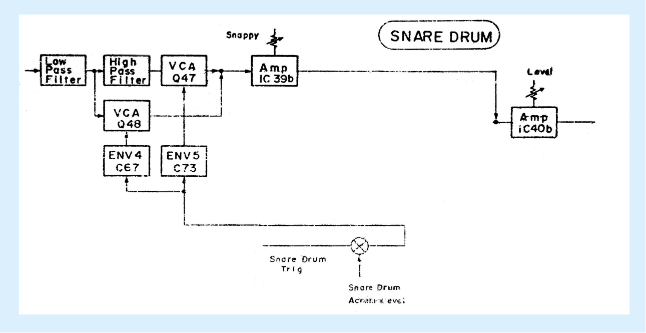


Figure 5: The noise signal path in the TR909 snare drum.

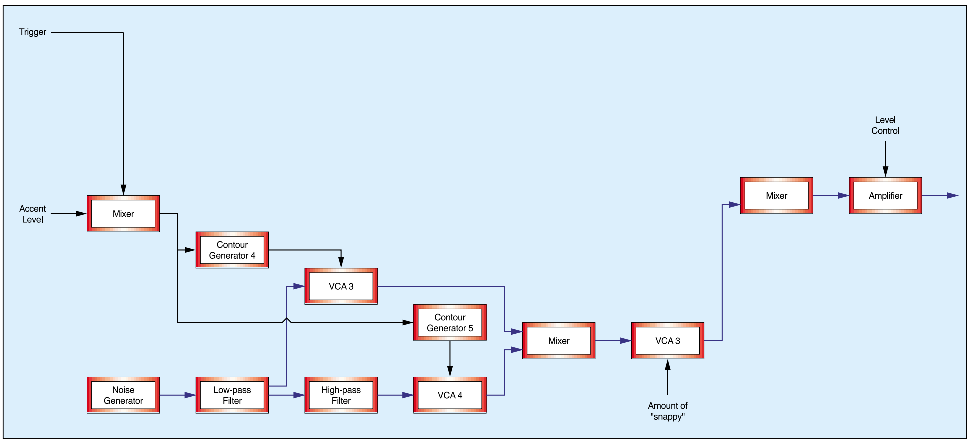


Figure 6: Figure 5 converted into Synth Secrets format.

## The TR808 Snare

Despite this, there's still no chance that you could patch Figure 7 on a

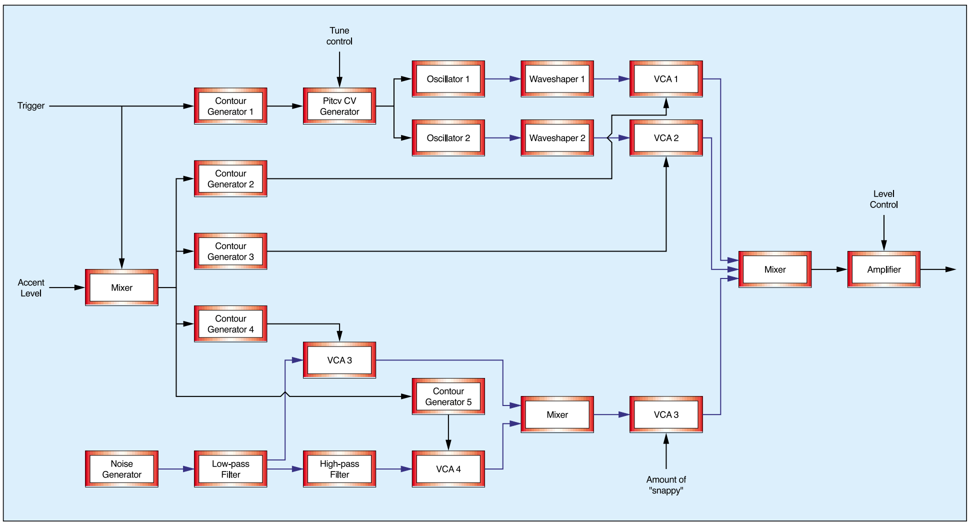


Figure 7: The Roland TR909 snare drum circuit laid out in Synth Secrets style.

Roland SH101, an ARP Axxe, a Minimoog, or any number of other common instruments. So is there an even simpler way to make passable snare sounds? Well, we'll have to take some short-cuts, but we need look no further than the Roland TR808 to find some of the techniques we need.

Figure 8 (below) shows the TR808's snare drum generator. This is much simpler than the TR909's equivalent, so let's see what's happening. To make things a little clearer, I've laid out Figure 8 in Synth Secrets style, making Figure 9 (right). This may look more complex than Roland's block diagram, but there is a reason for this: I have shown the trigger, the mixers and the external white noise generator.

As you can see, the trigger source has two jobs. Firstly, it fires a contour generator (left off the Roland diagram). The output from this (called the 'snappy' signal) is attenuated and added to the trigger itself, and the composite signal is then directed to both oscillators.

The oscillators, as you may have guessed, are of the 'bridged T' form that I described when I discussed the TR808 kick (see *SOS* February 2002 or www.sound-on-sound.com /sos/Feb02/articles/synthsecrets0202.asp). This means that you merely have to kick them into life, and they will then oscillate at the pitch, and decay at the rate, determined by their component values. The TR808 mixes the waves generated by the two oscillators in proportions that are again predetermined... this time by the factory values of the components in the mixer.

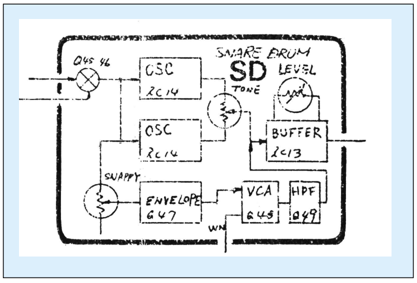


Figure 8: The Roland TR808 snare drum.

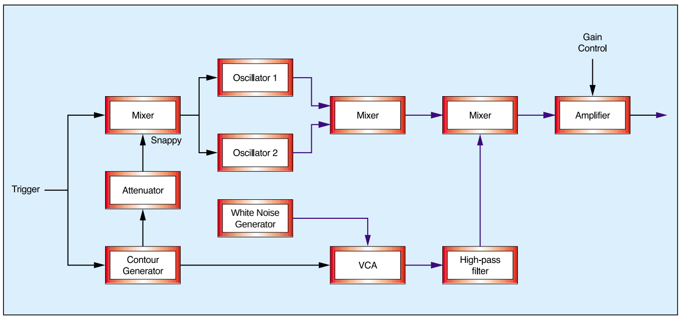


Figure 9: The TR808 snare diagram in Synth Secrets format.

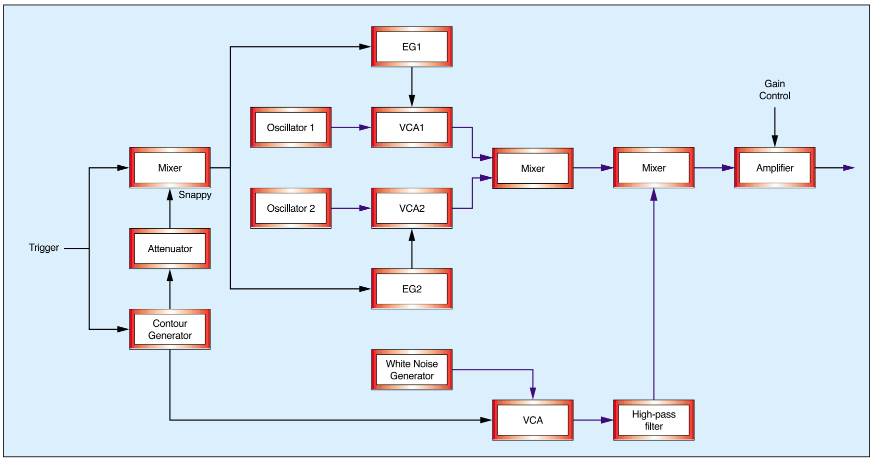


Figure 10: Replacing the bridged-T oscillators in Figure 9 with conventional VCO/VCA/EG modules.

Simultaneous with all of this, the contour generator is determining the gain of a VCA that is itself controlling the amplitude of a white noise signal. The output from this VCA is then high-pass filtered and mixed with the oscillators' outputs before the final composite passes to the output amplifier.

The beauty of this architecture is again in its simplicity and its low cost. Even if we replace the bridged-T oscillators with more conventional VCOs, VCAs and EGs (as in Figure 10 below), it remains straightforward. And again, it conforms closely to the principles that led me to create Figure 1... two decaying oscillators for the important 0,1 modes, plus a contoured noise source to emulate the sound generated by the presence of the snare itself. OK, the shifted harmonic series are missing, as are the holes in the noise spectrum, so the TR808 is never going to fool you into thinking that you're listening to a real snare drum. Nevertheless, it does capture the essence of 'snareyness'.

Now let's look again at Figure 10. To patch this we will need two independently tuneable oscillators, three VCAs, three contour generators, two audio mixers and a CV mixer, a noise generator and a high-pass filter. Well, we can save on one of the audio mixers by combining the two shown in series in the main signal path, but it's still a significant chunk of hardware. Yet our experience of many cheap and cheerful synths tells us that we can obtain snare-like sounds from even the simplest instruments. So how do we do it?

## Shapin' Some Noise

To answer this, we're going to travel even further back in time, to the 1970s, an era when drum machines (as they were then known) were ghastly little affairs that spat out rhythm sounds based on filtered noise. Static low-pass or band-pass filters determined the 'colour' of these sounds, and a VCA and AR contour generator shaped the signal amplitude (see Figure 11, below). This suggests that the simplest synths can generate recognisable percussion sounds, *provided that* they have a noise generator whose output can be treated by a filter and a contoured VCA.

Now, I don't know if there were any commercial drum machines that conformed to the diagram at the bottom of the page (see Figure 12) but a slightly more sophisticated unit might also have shaped the tone of the noise burst by contouring the frequency cutoff of the filter. If this looks familiar, I'm not surprised: with an extra couple of VCAs to control the amount of contour applied to the filter and the audio signal amplifier, it's the same as the block diagram I drew for the ARP Axxe and Roland SH101 a couple of months ago (see Figure 13, above right). So, let's program a snare sound on the Roland SH101, and see how close we can get to an authentic snare drum. Or, if that eludes us, let's see how close we get to the classic analogue snare sounds of the TR808 and TR909.

## Snares On The Roland SH101

We'll start by creating a patch based on Figure 12. Since we want no conventional waveforms in the sound, we go straight to the source mixer in the centre of the control panel to set the pulse wave, sawtooth wave and sub-oscillator contributions to zero. We then raise the noise level to 10, ensuring that the noise generator drives the rest of the signal path as hard as it can. The remaining controls that determine the source waveforms now become irrelevant, and we end up with the simple settings shown in Figure 14 above.

Now we must choose appropriate settings for the filter, and determine the amplitude contour. The noise burst generated by a real snare drum is rich in mid and high frequencies, so it is important that the synthesizer's low-pass filter is fully open for this sound. We achieve this by setting 'VCF Freq' to 10. However, the SH101 keeps a bit of cutoff frequency 'in reserve', so we can open it even more by setting the 'VCF Env' and 'VCF Kybd' sliders to 10, thus ensuring that the filter is at its most 'open' at the start of the sound, and that it closes slightly as the sound decays. This proves to be just what we want. However, it's important not to introduce any resonance ('Res') into the patch. This would make the sound go 'oowwww' as it decays, which would be quite inappropriate.

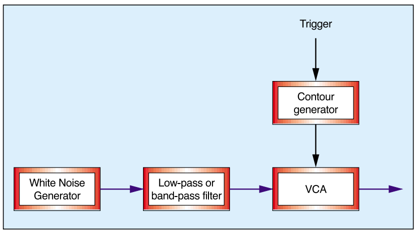


Figure 11: The simplest (and least authentic) way to create a snare drum sound.

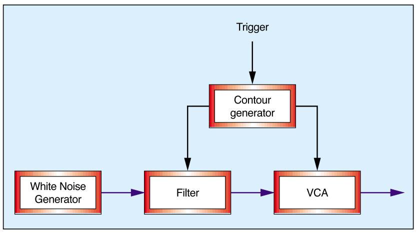


Figure 12: Almost the simplest way to create a snare drum sound.

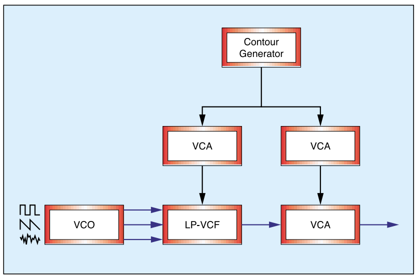


Figure 13: Some key elements of the ARP Axxe and Roland SH101.

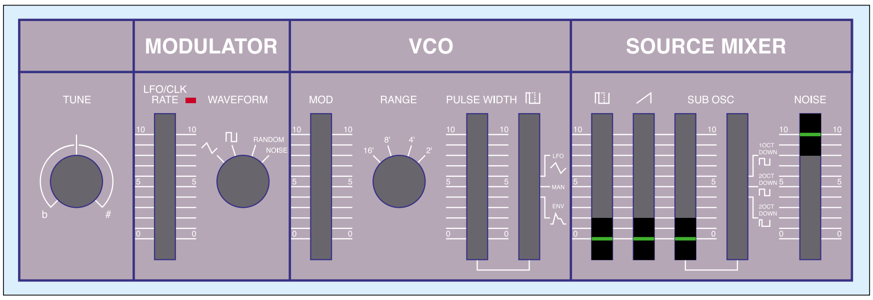


Figure 14: The initial waveform for our SH101 snare drum patch.

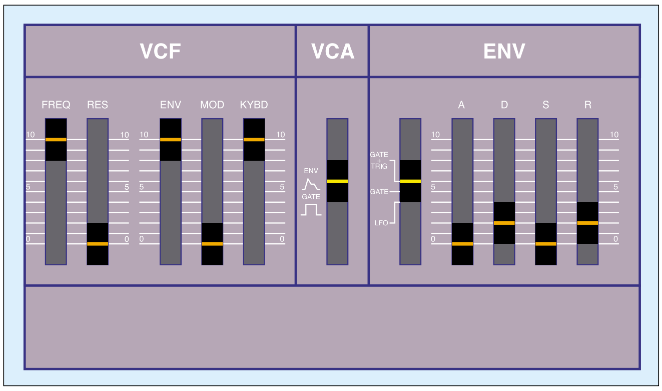


Figure 15: The filter and amplifier settings for our synthesized snare drum sound.

Moving on to the amplifier and envelope (contour) generator, we set the VCA to respond to the ADSR contour, and the keyboard triggering to 'Gate + Trig', which allows us to play fast snare drum rolls. Finally, we come to the ADSR envelope itself. The Attack should be as near-instantaneous as the synth can manage, so A=0. Likewise, a snare drum should never sustain, so S=0, too. This then leaves the Decay and Release times, which should be equal, ensuring that the sound decays consistently whether you release the keys or not. I find that setting Decay and Release times to '2' works just fine for this sound (see Figure 15 above).

Putting Figures 14 and 15 together then yields an acceptable analogue snare sound. Indeed, I find this patch to be far punchier that the early drum machines it emulates, for three reasons: the SH101 envelope is far snappier, the bandwidth is greater, and the noise generator slightly overdrives the filter input, contributing to a harsher timbre.

If you play this patch, you will be able to play single hits and snare rolls (by trilling on two adjacent notes). You can even imitate the inconsistencies with which a real drummer hits the batter head by reducing the 'Freq' setting (say, to 5 or 6) and playing a range of notes across the keyboard. With the keyboard tracking at maximum, you will find that notes at the top of the keyboard are brighter than those as the bottom. This adds a pleasing variation to the sound.

You can now EQ the result and add reverb to create a huge range of analogue snare sounds, from a booming deep-shell snare to bright military drums, to the gated excesses of everyone who copied Phil Collins throughout the '80s... you can obtain all of these from slight variations on this simple patch.

## The Sound Of The Drum

Now let's find out what happens if we try to introduce a tonal element into our patch, to more closely approximate the upper halves of Figures 1 and 10. These are the parts which attempt to generate the sound of the drum

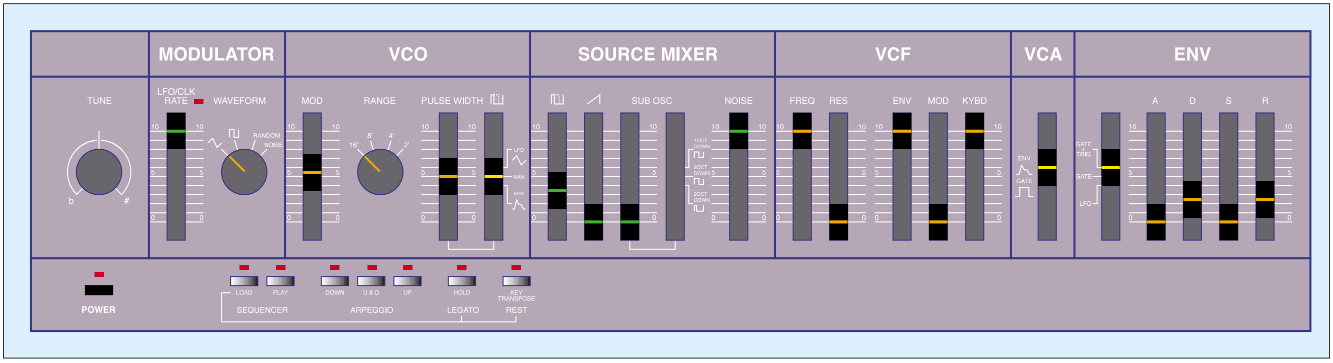


Figure 16: The SH101 'FM' snare drum.

itself, without its snare, as explained last month. Figures 1, 2, 3, 4, 7, 8, 9 and 10 all suggest that we need at least two oscillators for this, but the SH101 only has one. How can we use this to add realism to the sound?

Figure 13 shows that the SH101 can combine its pulse wave and/or the triangle wave with the noise generator. But, since they are just different waveforms produced by a single oscillator, they are at the same pitch. If we don't make the filter self-oscillate, the SH101 can't produce two pitches simultaneously. Or can it...?

Let's jump back to the left-hand side of the SH101 control panel. Apart from the master tune control (which is not of immediate interest) this is where we find the modulator. In general, we use this to generate vibrato (pitch modulation) or growl (frequency modulation). However, the maximum frequency of the SH101 modulator is just fast enough to encroach upon the bottom end of the audible frequency range, and this means that we can use it for frequency modulation, or FM.

Do you remember all of the horrid equations that you skipped in April 2000's Synth Secrets? (see www.sound-on-sound.com/sos/apr00/articles/synthsecrets.htm). Of course you don't... you skipped them. However, the ideas contained within them can be useful, as the following explanation might demonstrate. One of those equations (was it really two years ago?) described the series of frequencies generated as 'side-bands' when one sine-wave oscillator modulates the pitch of another. You may also remember (or more likely also skipped) the bit where I explained that, if the amount of modulation is low, only the first couple of side-bands are audible. The upshot of this simplification is that if you slightly frequency-modulate one signal (the carrier) with another (the resonator) the result is that you hear the original carrier and modulator signals, plus a couple of sidebands at the sum and difference frequencies of the carrier and modulator. If you're OK with maths, this can be expressed as the equation below, where '$w_{sb}$' is the frequencies of the sidebands, '$w_c$' is the carrier frequency, and '$w_m$' is the frequency of the modulator.

$$W_{sb} = W_c \pm W_m$$

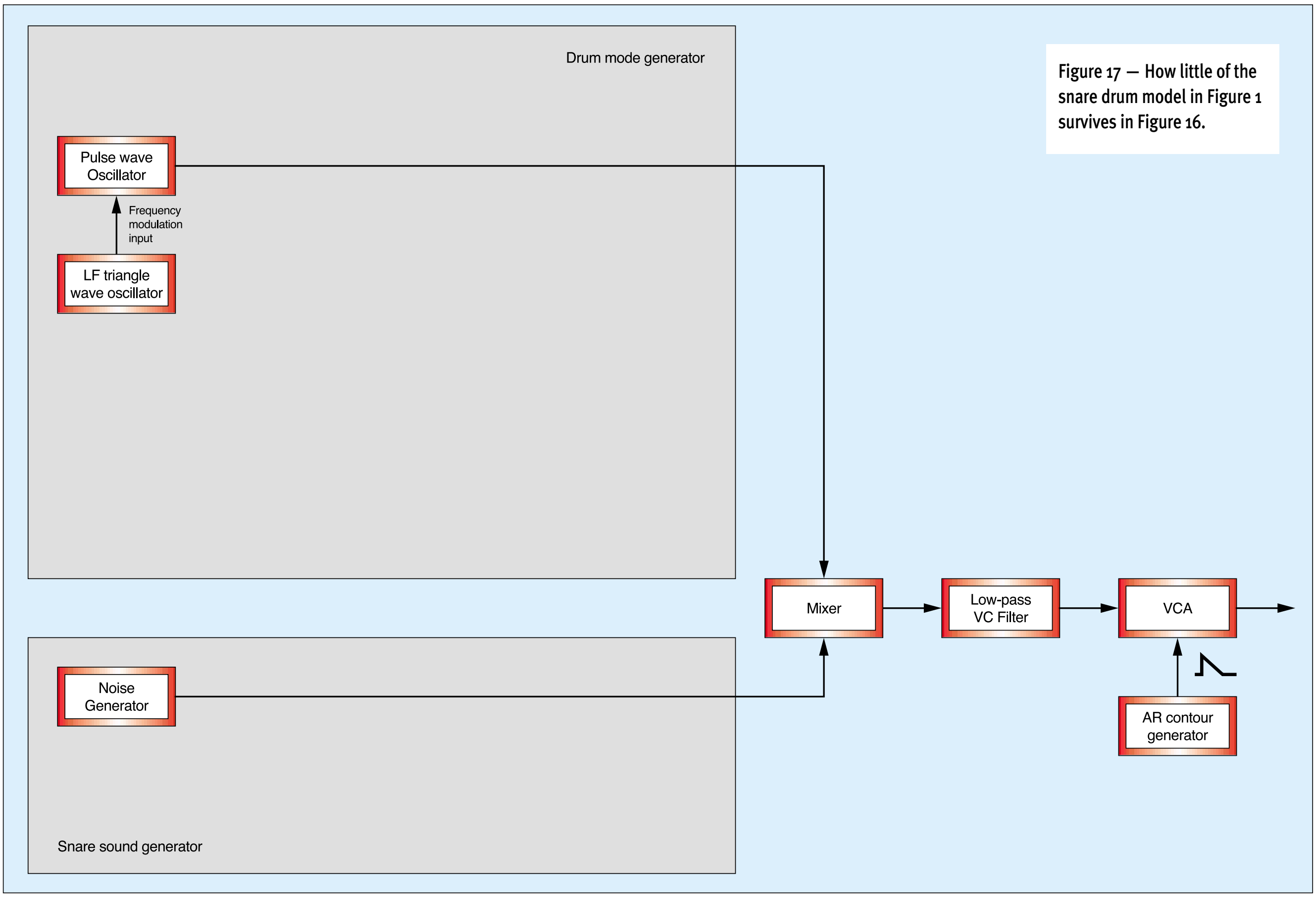


Figure 17 — How little of the snare drum model in Figure 1 survives in Figure 16.

Let's stick some numbers in here to make more sense of things. Imagine that the VCO carrier frequency is 200Hz. If the modulator frequency is 30Hz, we obtain no fewer than three audible frequencies in the output. These are the oscillator itself (200Hz), the oscillator *plus* the modulator (230Hz), and the oscillator *minus* the modulator (170Hz). Actually, 30Hz itself, as the modulator frequency, will also be present, but it's unlikely that you will hear this.

Now, these three frequencies look very different from the partial series that we discussed in last month's analysis of the snare drum. But I have ignored a crucial point: the SH101 does not generate sine waves. The modulator offers triangle and square waves (both of which have extended harmonic series) and the oscillator offers sawtooth and pulse waves (both of which *also* have extended harmonic series). This means that every harmonic in the modulator waveform interacts with every harmonic in the oscillator waveform, and the SH101 will generate a huge number of low-amplitude FM partials.

Of course, the frequencies thus generated (and their amplitudes) are quite different from those generated by the shell of a naked snare drum. On the other hand, the drum's modes are so screwed up by their interaction with the snare mechanism that FM is as likely as any other method to generate an acceptable imitation of the shell.

So, it's time to get empirical: that is, to experiment to find the settings for 'LFO Rate', 'Mod', and VCO frequency that give you the type of 'shell' sound you're looking for. I happen to like the settings in Figure 16 (top left) played on the uppermost 'G' of the keyboard. I have set the LFO rate to maximum, the 'Mod' amount to a middling value, and chosen a 25-percent pulse wave to be the carrier. I have then added about 35 percent of the resulting sound to the noise in the mixer. If these settings seem overly precise, they're not. This patch is very sensitive to tiny changes in the Mod and pulse-width settings, so — quite apart from personal preferences — the small differences between synths should guarantee that your patch will sound different from mine.

We'll finish by encapsulating the simple patch in Figure 16 as a block diagram, laid out in the same way and on the same scale as Figure 1. Figure 17 (below left) shows how much (or rather, how little) we have retained from our original ideas. As you can see, most of this diagram is empty, and as well as losing control over the precise nature of the sound spectrum in the drum mode generator, we've lost all the independent VCAs and contour generators that allowed us to shape the various parts of the sound. And that is perhaps the most significant difference between a 'real' synthesized snare drum, and the analogue sounds that people use instead of snares in much of today's music.

And that's about it for this month... Sure, we could discuss in greater depth why we can dispense with some of the elements required by our theoretical analyses, but not others. I could also demonstrate how we can use the same principles to create similar sounds on synths such as the Axxe or the Minimoog, or even show you how to use the multiple filters and modulation routing on the Korg MS20 to get much closer to the sound of a real snare. But I reckon that, with all the theory from last month, plus the explanations of the TR909 and TR808, and this SH101 example, you should now be in a position to create some excellent snare drum patches of your own. So get twiddling... SOS

# Synth Secrets

## Analysing Metallic Percussion

Original photo courtesy of Zildjian

Gordon Reid

**The task of synthesizing convincing metallic percussion defeated many synth giants — you only need to listen to Kraftwerk's weedy cymbals on 'The Model' to be persuaded of that. So why is it so difficult? We find out...**

In this month's Synth Secrets I'll begin my examination of the next family of important percussion instruments we're going to try to synthesize — idiophones constructed from thin metal sheets. In other words, cymbals, hi-hats, tam-tams and gongs.

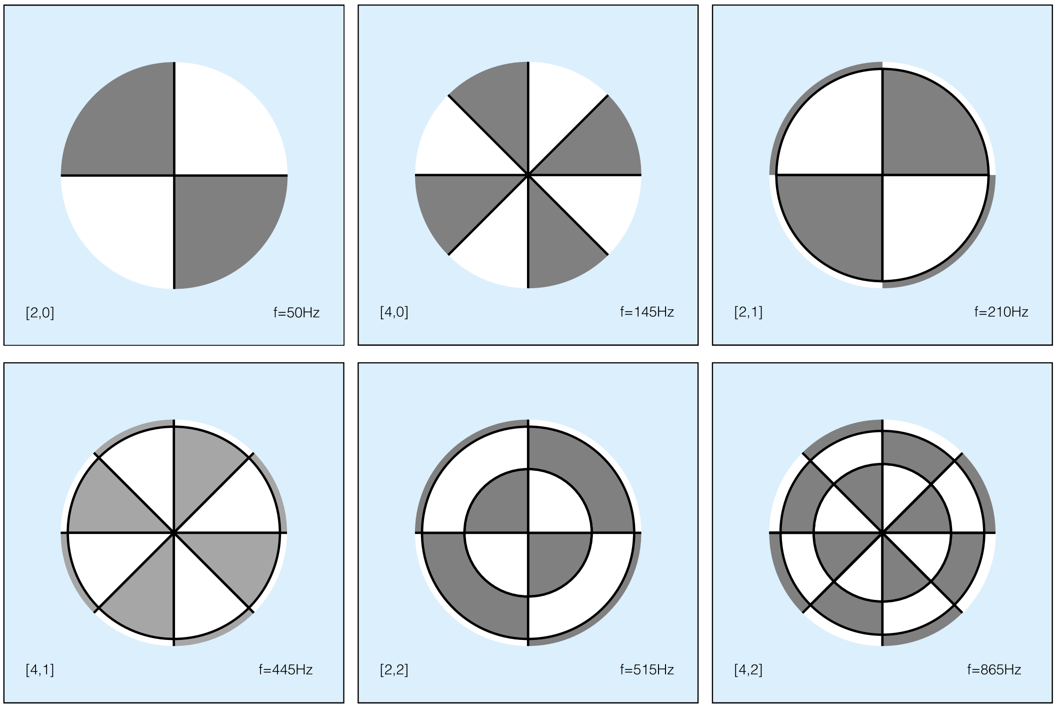
Before we can begin any synthesis of cymbals and related instruments, we need to understand something about the vibrations that occur in a circular metal plate. Don't worry, this isn't a prelude to something that looks like an excerpt from a university textbook because, as I have been forced to write on a number of occasions in recent Synth Secrets, a full analysis of this is *far* beyond the scope of these articles. If you want to frighten yourself, say after me, "the equations of vibrational motion for circular plates are the solutions to combinations of Bessel functions and Hyperbolic Bessel functions". I don't know about you, but that scares the daylights out of me. So, as we found two months ago when we discussed the sound-generation system of the snare drum, we will fall back on observation and measurement to gain a modest understanding of this month's topic: the class of cymbals that includes the common crash and ride cymbals.

### Simple Modes

As with so many of the instruments I've been trying to synthesize recently, there's a lot more to a cymbal than meets the eye. Firstly, there's the material to consider. It's no accident that cymbals are made from a narrow range of alloys, because these have proved to provide the most sonorous tones.

Secondly, there's the geometry of the cymbal: its size, thickness and shape. The cymbals used in rock music sound quite distinct, and few people would mistake a ride cymbal for a full-sized crash. And few listeners would confuse domed cymbals with the sounds produced by tam-tams or gongs, which have a turned up rim.

Thirdly, there's the manner in which you play them. Nowadays, we tend to think of cymbals primarily as instruments that have the living daylights beaten out of them using wooden sticks. Occasionally, though, you might see them played using softer beaters or mallets, and in orchestras it's common to see two cymbals crashed together by hand... hence the name, crash cymbals. But how frequently do you see cymbals played using a bow? Not often, although bowing a metal plate was once



Figures 1a to 1f (top left to bottom right): Six of the simpler modes of a flat circular plate.

a recognised way of producing a sustained tone.

So let's start by considering what a cymbal is. It's no good saying that it's a sheet of beaten metal with a bump in the middle or a turned-up rim. Sure, that's frequently true, but it doesn't get us very far. What's more, we can't treat it as some sort of metal drum head, because there's little that links the two, despite superficial similarities such as physical shape. The circular drumhead we've been discussing for the past few months is clamped (and therefore unable to move) at its edge, but it is free to oscillate elsewhere in a manner determined by air pressure and its own tension, thickness and stiffness. A circular metal plate mounted firmly at its centre is unable to move at that point, but is free to vibrate at its rim. As you might expect, the types of motion each undergoes are significantly different.

Furthermore — again unlike the drumhead — the plate is a rigid structure. This means that it is subject to the physics of solid objects, with all manner of vibrations propagating within the body of the metal itself. All in all, it's small wonder that plates sound so different from drumheads.

Figures 1a to 1f (below left) show six of the simplest (!) modes of vibration for a flat circular plate of uniform thickness, calculated using the hideously complex equations mentioned earlier. In these diagrams, a stationary position appears black, with the grey areas 'up' and the white areas 'down' (or *vice versa*) at any given moment. This leads to an interesting observation: not only can the rim of the plate move up and down, it does so in all the modes shown, and, if the academics are to be trusted, in all the others too.

In principle, it's not hard to observe the natural modes of vibration of a flat plate. If you energise it using a transducer that is itself driven by a sweepable oscillator, you will swiftly find that specific frequencies cause the plate to resonate, while others do not. If the plate is perfectly flat and horizontal, you can then use fine grains such as sand to observe the nodes. For example, if you scatter the sand evenly over the plate in Figure 1 and then energise it at 445Hz, you would find that all the sand congregates on the black lines shown in Figure 1d, sitting on the four radial and one circular node that define the 4,1 mode.

## Vibrations & Lasers

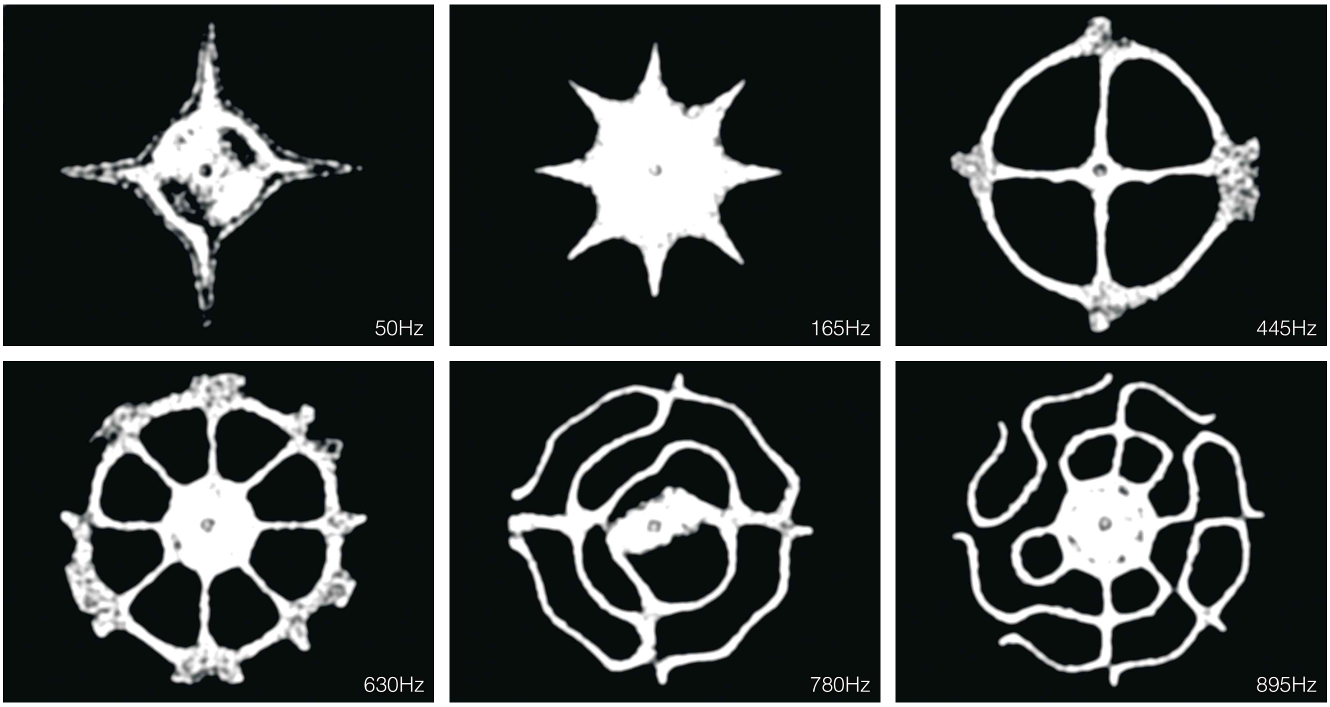
Unfortunately, determining the shape of the vibrations of a real cymbal is another matter. It's not flat, and very little of its surface is horizontal, so all the sand falls off. We need something a bit more sophisticated. Scientists observe the modes using a technique called holographic interferometery. In short, this requires that you split a laser beam, bounce one of the resulting beams off the vibrating surface, and then recombine the two beams on a holographic plate. Because the light from the two paths interferes when they are recombined, they create a pattern on the plate which, when viewed with another laser, creates an image of the stationary areas on the cymbal's surface. It sounds tricky, and it is. Nevertheless, it works, as Figures 2a to 2f — obtained by Dr Thomas Rossing at the Physics Department of Northern Illinois University — prove (see below).

The patterns making up Figure 2 show the six modes calculated for Figure 1 but, because the cymbal has a dome in its centre and exhibits inconsistencies in its thickness and rigidity, the images are not precisely the same as the theory of flat plates would predict. What's more, although interferometery shows the stationary points on the cymbal's surface (the white areas), it does not show which areas are 'up' and which are 'down'. Nor does it show the edge of the cymbal itself. Nonetheless, the patterns in Figure 2 are recognisably the same modes shown in Figure 1.

Looking again at these diagrams, you can now see how much the central dome of the cymbal influences its vibration. What is even more interesting is that the amount of the dome that remains stationary differs very markedly from one mode to another. It has — for example — a huge effect on the amount of metal that is free to move in the 4,0 mode, but much less effect on the 2,1 mode. You might also notice that, as the vibrations get more complex, the neat patterns resembling the top view of a Terry's Chocolate Orange break down, and numerous complex regions appear.

If you now study Table 1 (overleaf), which, as an example, compares the modes of vibration of a flat plate and a domed cymbal, both with fundamental frequencies

▶



Figures 2a to 2f (top left to bottom right): The same six modes for a western cymbal.

of 50Hz, the you can see that, like those of the flat plate, the frequencies produced by a real cymbal are *far* from harmonic (note that the frequencies in Figures 2(a) to 2(f) are not the frequencies observed by Thomas Rossing; I have adjusted them so that you can make a direct comparison with the calculated modes in Figure 1). What's more, there seems to be no simple relationship between the plate and the cymbal, other than that — for the same fundamental frequency — the cymbal produces consistently higher frequencies than the plate. Well, there *is* a relationship, but it is so complex that we have no choice but to ignore it here.

There can be hundreds of energised modes in an excited cymbal but, unlike the snare drum that we studied two months ago, the cymbal's modes exist at discrete frequencies, so we do not obtain a noise spectrum as we did with the snare. Hmm... that's not strictly true, as you will discover as you read on, but it will do for now.

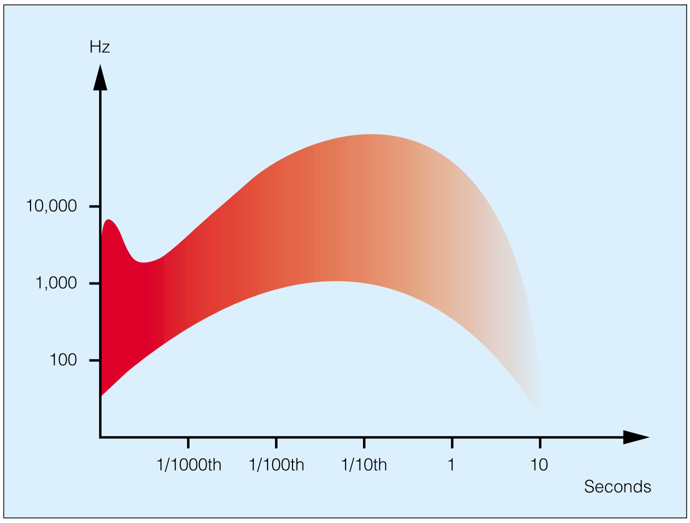


**Figure 3:** A rough representation of the way in which the cymbal sound develops.

| MODE | CALCULATED FREQUENCIES FOR FLAT PLATE (HZ) | OBSERVED FREQUENCIES FOR CYMBAL (HZ) | INCREASE IN FREQUENCY (PERCENT) |
|---|---|---|---|
| 2,0 | 50 | 50 | — |
| 2,1 | 210 | 445 | 112 |
| 2,2 | 515 | 780 | 51 |
| 4,0 | 145 | 165 | 18 |
| 4,1 | 445 | 630 | 42 |
| 4,2 | 865 | 895 | 3 |

**Table 1:** Comparing the modes of a flat plate and a domed western cymbal.

## What Happens When You Hit It?

If cymbals produced a unchanging combination of the modes described above, they would be very different from the instruments we know in the real world. However, the cymbal's sound is far from static, so we must now consider what happens when we hit one of them.

As drummers and percussionists are aware, the sound of a cymbal varies widely depending upon where you hit it, how you hit it (a downward stroke, a glancing blow, and so on...) and with what. One struck close to its edge with a soft beater or mallet will produce a very different sound to one pinged close to the dome using a stick. So there's no single way to describe the sound produced, and we will concentrate on a single instance: that of hitting the cymbal part-way between the edge of the dome and the edge of cymbal itself.

We'll start by forgetting about music for a moment, and by considering what happens when we throw a stone into a circular pond. We know from experience that the impact of the stone energises the surface of the water, and we see a circular wave propagate outwards until it hits the shore. If the stone is large enough, and the pond has hard edges rather than a gentle bank, we may even see the waves bounce off the edge, whereupon ripples will move back into the pond.

This analogy is not far divorced from what happens in the first few microseconds after we strike a circular plate. Indeed, if we make the pond shallower at its edges than its centre, we could extend the analogy to describe a plate of varying thickness, much like a real cymbal. However, it breaks down when we try to add the central dome of the cymbal, so we'll take it no further.

When the stick strikes the cymbal, the energy of the impact dissipates much as I have described in my watery analogy, propagating outwards at speeds of up to 4000 miles per hour, producing waves that bounce off the edges and the central dome, and which interfere with one another in an amazingly complex fashion.

This phase lasts just few milliseconds, while the energy converts itself into strong modal peaks at a few hundred Hertz. Then, over the next few hundred milliseconds, the energy moves upwards through the spectrum, and the cymbal rings at a few kHz. The high-frequency modes are the shortest-lived, however, so the last of the cymbal's energy lies in the modes at a few hundred Hertz, and — at the end — it's the mid-frequencies that again dominate the sound.

I have drawn a coarse representation of this in Figure 3 (left), in which the red band shows the band of frequencies that are most apparent to the listener. Note that I have used logarithmic axes for this figure, so the first three time divisions occur in a fraction of a second.

If this analysis were not complex enough, recent research suggests that it applies only to cymbals hit lightly. Indeed, if you tap one gently and then place your ear very close to the cymbal, you can hear these modes ringing in a beautifully metallic fashion. When hit harder, the modes split into patterns of even greater complexity, and produce sub-harmonic frequencies. However, there's an even greater complication... Research now suggests that, at high amplitudes, a cymbal's vibrations become chaotic. This means that we can no longer see the neat patterns of modes on the cymbal's surface, and the clearly identifiable modal frequencies disappear. At this point, the cymbal's spectrum is — in essence — noise. It's unlikely that the designers of early analogue drum machines knew this, because holographic interferometery did not exist at the time, but it suggests that they were not completely wrong to use white noise as the starting point for their basic cymbal sounds.

Armed with this knowledge, we can now synthesize a cymbal — but the details of how to go about this will have to wait until next month, when we'll see how we might attempt this on a modular synth. We'll also consider how analogue synth manufacturers took a simpler approach in the '70s to achieve roughly the same results.

**Thanks to Dr Thomas Rossing at the Physics Department of Northern Illinois University, USA, for his permission to derive Figures 2a to 2f from diagrams published in his research, and reproduce them here.**

# Synth Secrets

Having analysed the complex process by which a cymbal makes its sound, it's time to synthesize our own...

## Synthesizing Realistic Cymbals

Original Photo courtesy of Zildjian

Gordon Reid

Last month, we began looking at the complex way in which a cymbal produces its sound. We considered some of its modes of vibration via the wonders of holographic interferometry, and also analysed how the various modes develop after a cymbal is struck (broadly summarised in Figure 1, shown below).

Using this information, it's now possible to attempt to synthesize a cymbal. For reasons of simplicity, I have chosen to try creating a ride cymbal, because, to my ears, this class of cymbals produces less complex timbres than either crash or splash cymbals. What's more, its shorter duration makes it simpler for me to fool the ear into hearing what I want it to.

### The Initial Waveform

I'll start by deciding how to create the initial waveform. We know from last month's studies that that the cymbal's sound is a dense fog of enharmonic partials, and that it is not dominated by any particular modes. The easiest way to produce these is by using FM synthesis. If I take an unfiltered pulse wave and use it to modulate an unfiltered square wave at high amplitude, I can generate hundreds of partials across the whole bandwidth of the synth I use, as explained in Part 12 of this series (see *SOS* April 2000, or head for www.sound-on-sound.com/sos/apr00/articles/synthsecrets.htm).

At this point, it's important to acknowledge that the cymbal's spectrum extends above the normal limit of human hearing (for more on this, see the box on supersonic research opposite), and that its energy distribution is fairly flat. This means that I need to produce a flat (-ish) spectrum of FM partials, and must choose my modulator and carrier frequencies carefully to attain this.

Many analogue synths are incapable of satisfying this requirement; some exhibit a maximum cutoff frequency of just 12kHz or thereabouts, which (ignoring any other limitations) makes them unsuitable for the job at hand. So I have decided to use a digital synth with a full 20Hz-to-20kHz response to create the cymbal patch — Clavia's Nord Micro Modular (reviewed in *SOS* July 2000, or see www.sound-on-sound.com/sos/jul00/articles/clavia.htm). This software/hardware combination imitates a modular analogue synth very well, and is an ideal vehicle for illustrating the principles discussed.

Figure 2 (above right) shows the oscillators I have chosen for the patch. As you can see, I have directed the output from the modulator ('OscA1') to both the pitch input and the FM input of the carrier ('OscA2'), and set the amplitude of the modulation to maximum in both cases. Using both the Nord's linear and logarithmic modulation inputs ensures that the resulting spectrum is suitably complex.

You will also note that I have selected a pulse wave set to (approximately) 1kHz for the FM modulator and a square wave set to (approximately) 2.5kHz for the FM carrier. You might wonder why I have chosen these waveforms and frequencies, and there is a simple answer... it sounds suitable to me. A more analytical approach would suggest that this choice produces the required spectral width, with just the seventh side-band of the fundamental frequencies lying at 18.5kHz. Furthermore, this choice does not concentrate clumps of partials into narrow bands of frequencies, which is probably a good thing.

Clavia's Nord Micro Modular, as used in this month's cymbal synthesis exercise.

### Shaping The Sound

We must now shape the output from the carrier to create the sound we want. As Figure 1 suggests, we'll do so using two signal paths: one that passes an initial burst of mid-frequency partials to imitate the initial impact, and a second that produces the decaying high and mid frequencies in the extended tail of the sound.

Figure 3 (opposite) shows the architecture I have used to generate the initial 'ping' of the cymbal. This passes the carrier's output through a 24dB-per-octave band-pass filter. I have set the base frequency of this to be approximately 1kHz, and then applied a fast Attack-Decay envelope that sweeps the pass-band down to 1kHz over the course of just one-fifth of a second. The attack is as near-instantaneous as it is possible for a Nord envelope to be.

I have used the same envelope to shape the amplitude of the sound, taking it from maximum amplitude to silence over the same period (because, as explained last month, all the energy in the 'ping' is translated into

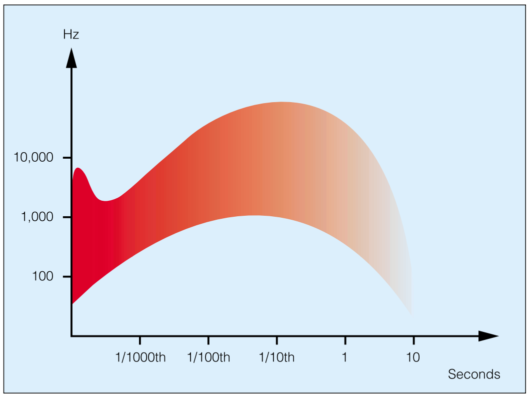
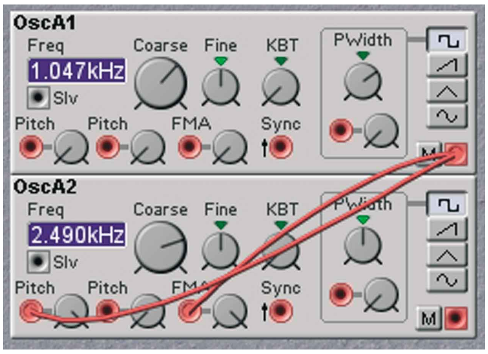


Figure 1: A rough representation of the way in which the cymbal sound develops, as first shown in last month's article. Note that the scale of the horizontal axis is logarithmic, not linear.



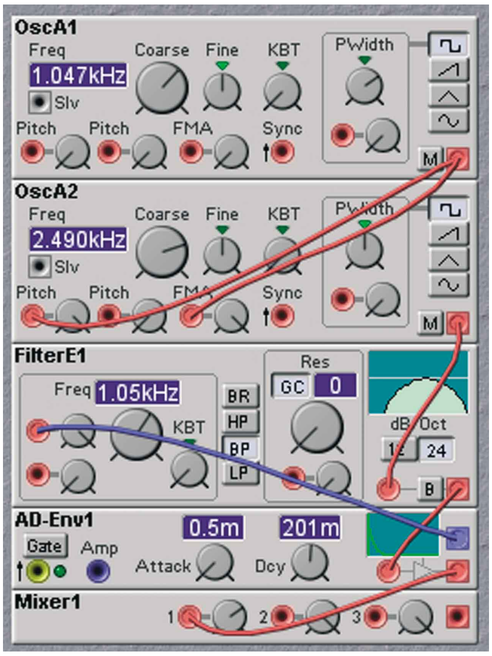
OscA1 Freq 1.047kHz

OscA2 Freq 2.490kHz

Figure 2: Creating the basic signal for the cymbal patch.

higher-frequency modes in the first few milliseconds). The output from this part of the patch is then directed to input 1 of the 'Mixer1' module.

At the same time, I have passed the carrier's output through a 12dB-per-octave high-pass filter to generate the tail of the sound. This filter (shown on the next page in Figure 4 as 'FilterD1') has a basic frequency of 2.64kHz, and the Attack-Hold-Decay envelope assigned to it ('AHD-Env1') opens it from this setting to its maximum over the course of approximately 200 milliseconds. This emulates the way in which the energy of the initial impact generates the higher partials. The filter then closes slowly — I have chosen a setting of 3.7 seconds — returning to its basic cutoff frequency.



OscA1 Freq 1.047kHz

OscA2 Freq 2.490kHz

FilterE1 Freq 1.05kHz, Res GC 0

AD-Env1 Attack 0.5m, Dcy 201m

Figure 3: Creating the initial 'ping'.

Having created the two signal paths, I have combined them as shown in Figure 5 on the next page, using the 'Mixer1' module to mix the elements in suitable proportions. You will note that I have given the tail a greater amplitude than the 'ping', and again this is simply because it sounds correct.

The final element in the audio path is a further Attack-Decay envelope generator ('AD-Env2') that shapes the combined sound. This has a near-instantaneous attack, and a decay of around three-quarters of a second, which seems about right for the cymbal we're emulating. You may ask why the tail of the sound produced in Figure 4 is so long if it is then truncated so severely by the 'AD-Env2' module. I have found that this produces a natural envelope when you re-trigger the sound: longer gaps between notes allow the pass-band to drop to lower frequencies than short gaps do, and this introduces a subtle but natural variation into the sound. Again, this seems to make sense, but it would probably take too long to analyse why it should be so.

Now, rather than play the patch from a keyboard, I have chosen to drive it using a

▶

## Do Supersonics Matter?

**A researcher at the California Institute of Technology, James Boyk, has measured the frequency spectrum of a crash cymbal, and found that, unlike that of other instruments, the cymbal's spectrum "shows no sign of running out of energy at 100kHz". He has also demonstrated that fully 40 percent of the sonic energy produced by a cymbal can lie above the commonly accepted threshold of human hearing at 20kHz. This raises two interesting questions. Firstly, if, as is generally accepted, most humans are unable to hear sounds above 20kHz, does the cymbal's supersonic energy still affect the audible sound in some way? Secondly, if it *does*, do we need synths, mixers, amplifiers and speakers with bandwidths exceeding 100kHz to reproduce these sounds accurately?**

**It's unlikely that anyone will be answering these questions at any time soon, but Boyk's research still has an important consequence when synthesizing cymbals: we must be able to create flat frequency spectra up to the accepted limit of human perception, if not beyond. This imposes constraints upon the type of synth we can use.**

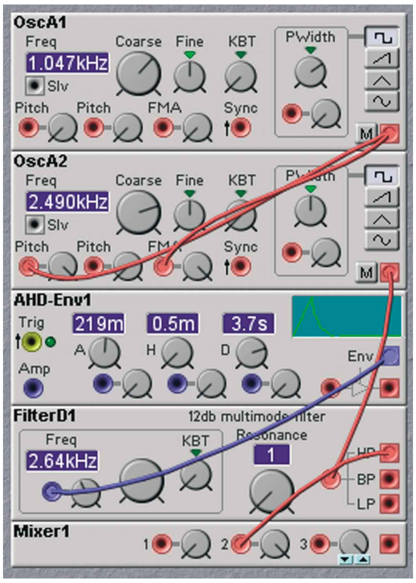


Figure 4: Creating the high-frequency tail of the sound.

sequencer clocked by an LFO. You can see these in the upper-left corner of Figure 5. If you look closely, you can see that I have programmed a six-beat sequence with the cymbal triggered every first, fourth and sixth beat. This produces the classic 'tscccchhhh t-t tscccchhhh t-t tscccchhhh' so beloved of jazz and swing drummers, which is a perfect test for a ride-cymbal patch.

So, how does it sound? To be honest, the result is not as realistic as I had hoped, largely because the initial FM spectrum is not quite close enough to the real thing. Instead, this patch is not unlike that obtained from a high-quality analogue drum machine. It's recognisably not a real cymbal, but it does exude what you might call 'cymbal-ness', and would no doubt work well in a mix. What's more, it achieves this without recourse to a noise generator, and produces better results, to my ears at least, than those from the drum machines that use noise for cymbal sounds (of which more next month). Finally, I think it's an improvement on the drum oscillator provided within the Nord architecture.

Figure 6 (below) now shows the sound-generating modules from Figure 5, but drawn in standard Synth Secrets format, so that you can apply this patch yourself to something other than the Nord Modular. You can see the dual signal paths, with the upper path creating the 'ping', and the lower the extended tail of the sound.

Of course, this is not the end of the story — in fact, it's the beginning. Given time and patience, I'm sure that I could create a far more realistic ride cymbal on the Nord. It might require more signal paths, or even complete new synthesis mechanisms including further oscillators as well as filters and so on... but I'm confident that it could be done. Unfortunately, this would take us so far beyond the capabilities of typical analogue synths that it would be meaningless for anybody lacking a Nord Modular, Native Instruments' *Reaktor*, or about £25,000 of extended Moog System 55.

More realistically, we could adjust the parameters of this patch to generate staccato timbres that sound like a closed hi-hat, or to create open hi-hats and other types of cymbal. And, of course, there's nothing stopping you creating similar (or better) patches on suitable analogue (or virtual analogue) synths.

But what if you don't own a big modular synth — analogue or digital? Does this mean that cymbal sounds are forever beyond your reach? Next month, we'll look at Figure 6 in closer detail and, as we have done before, see whether we can simplify the patch sufficiently to create convincing cymbal sounds on low-cost analogue synths. Until then... SOS

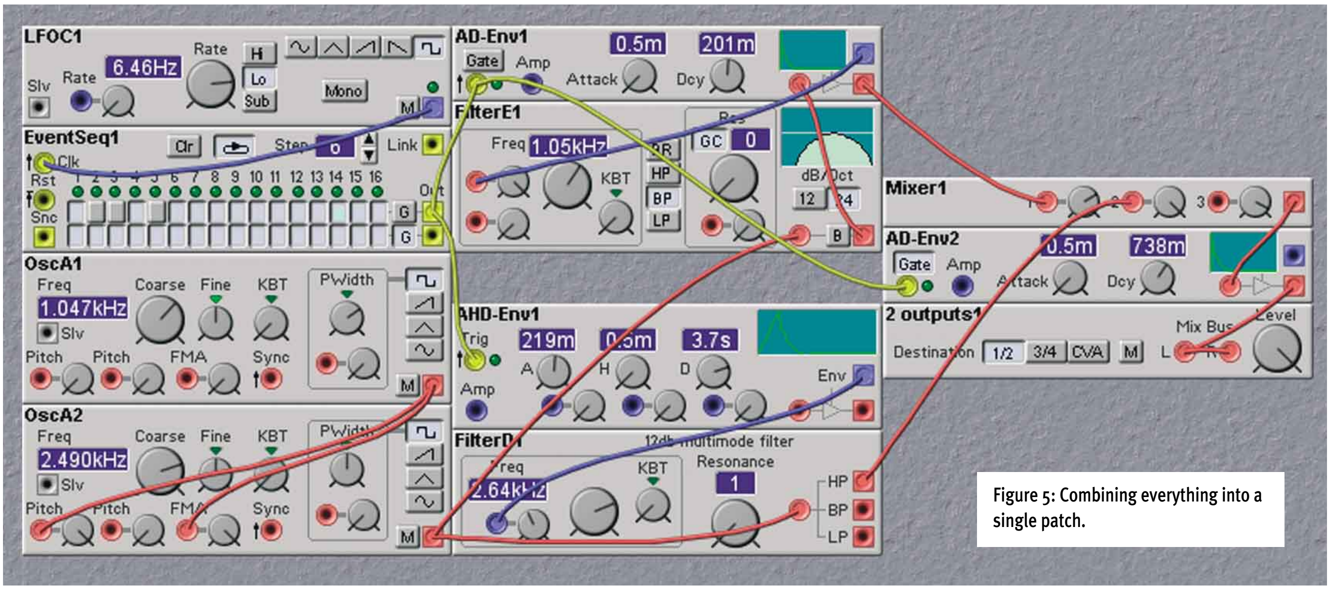


Figure 5: Combining everything into a single patch.

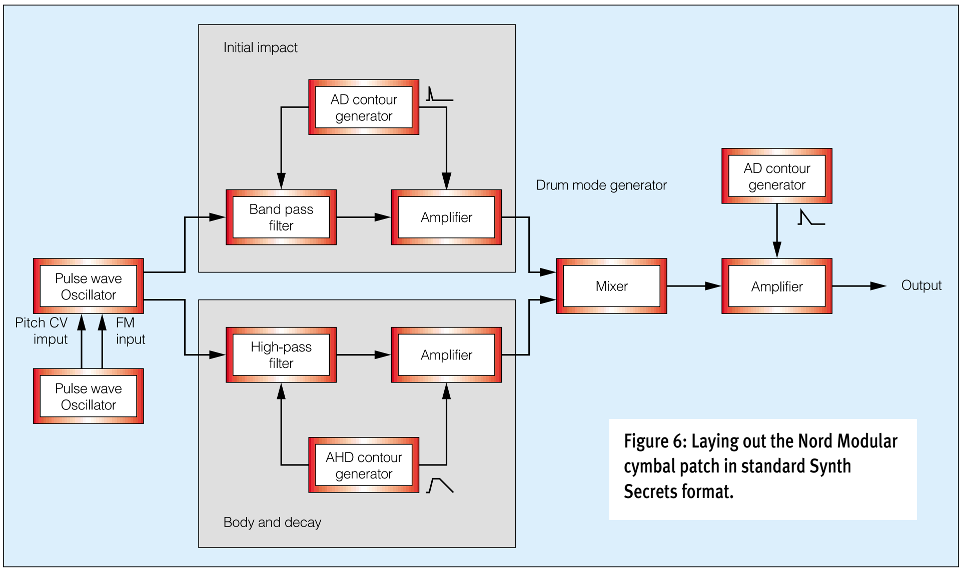


Figure 6: Laying out the Nord Modular cymbal patch in standard Synth Secrets format.

# Synth Secrets

## Practical Cymbal Synthesis

**Synthesizing realistic cymbals is complex, but not impossible — after all, over 20 years ago, Roland's TR808 drum machine featured synthesized cymbals. We look at how they managed it, and attempt to create cymbals on another affordable analogue synth.**

Original Photo courtesy of Zildjian

Gordon Reid

Three months ago, in the May issue of *SOS*, I analysed and dissected the snare sounds produced by the Roland TR808 and 909 — which, by common consent, are two of the 'classic' electronic drum sounds (see www.sound-on-sound.com/sos/apr02/articles/synthsecrets0402.asp). So, following the analysis of the cymbal in last month's *Sound On Sound*, I thought that it made sense to do the same this month for the cymbal sounds produced by both of these vintage drum machines. Of course, the cymbal sounds don't have quite the same hip *je ne sais quoi* as the snares — but surely if Roland were able to synthesize cymbals in the early '80s, then despite all the complexity I detailed last month, it can't be that hard... can it?

### The TR909 Cymbal

Figure 1, above, shows the cymbal patch I developed last month on my Nord Micro Modular, and Figure 2 (right) shows a simplified schematic for the TR909 cymbal sound generator. As you can see, they are utterly dissimilar. The key to this difference lies in the bottom left-hand block of Figure 2, the one that says 'six-bit data table'. Forget analogue FM synthesis, dynamic band-pass filters, and all the other paraphernalia I employed to try to recreate the cymbal's complex spectrum — the TR909 dispenses with all of this by incorporating a digital sample of a genuine cymbal. In other words, at least one (and, in truth several more) of the instruments in the TR909 are generated by a 'samploid' drum machine. This means that parts of the TR909 are little different in concept and implementation from any other synth with PCM samples as oscillators.

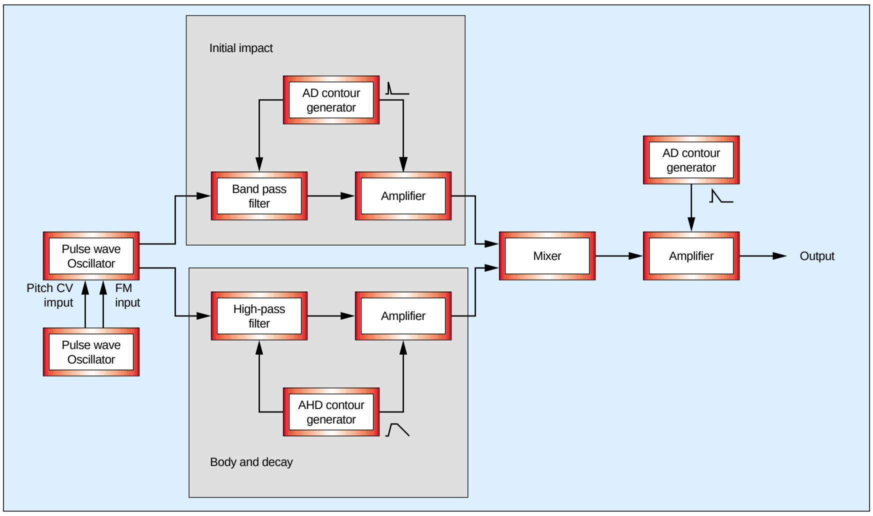


Figure 1: A simple, subtractive model for the cymbal sound.

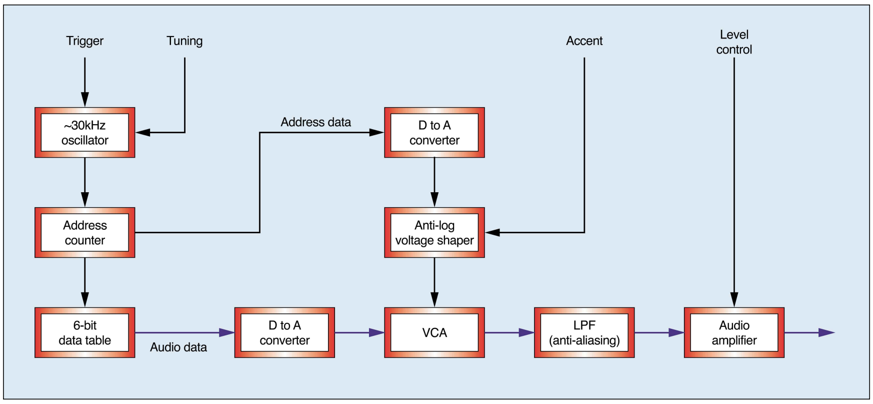


Figure 2: The TR909 cymbal sound generator.

The reason for Roland's decision is simple and compelling. It is *very* difficult to emulate a cymbal using analogue synthesis. You might well have guessed this already if you were following the ins and outs of last month's article, but if you need convincing further, I will demonstrate the point more clearly later. Before that, though, let's see what's going on in Figure 2. Although we've largely avoided digital electronics in Synth Secrets, we should be able to understand the principles without too much trouble.

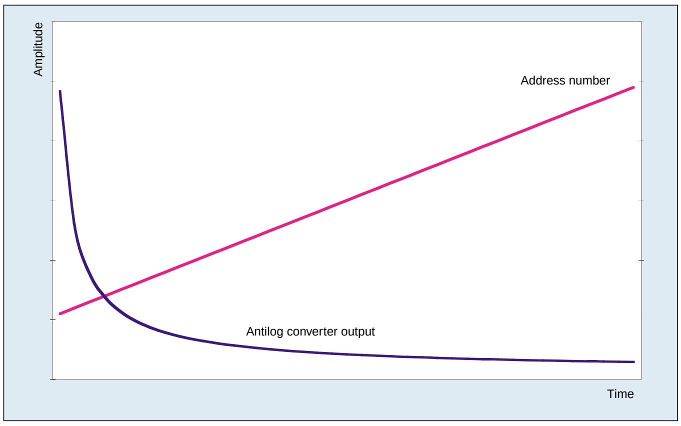


Figure 3: Turning an address into a contour.

We'll start with the 30kHz oscillator to the upper left of the diagram. This is a clock that, when triggered, causes an address counter to revert to 'zero' and then step through the memory addresses of the sample data held in the ROM.

There are two outputs from the address counter. The first provides the information required by the TR909 to replay the audio data from memory. These exist in digital form, so the samples pass through a primitive digital-to-analogue converter (DAC) and the analogue signal produced by this then passes to a VCA for amplitude shaping. This is the audio path that runs horizontally along the bottom of Figure 2.

The second output from the address counter contains the address data itself. To understand this, imagine that the first five samples in the data table contain numbers that we will call 'v', 'w', 'x', 'y' and 'z' (the actual values are not important). The numbers are held in specific locations in the table that we will call '1', '2', '3', '4' and '5', because v, w, x, y and z are the first five samples.

Now look back at Figure 2. As the audio values (v, w, x, y and z) are sent to their DAC, the numbers of the addresses that contain them — in this case, 1, 2, 3, 4 and 5 — are directed to a second DAC at the top of the diagram. The voltage produced by this DAC then passes through a device called an anti-log converter which takes the increasing voltage produced by the DAC, and turns it into a suitable envelope that controls the gain of the VCA in the signal path (see Figure 3, above).

At this point, you should ask why the TR909 employs such a complex mechanism to replay its cymbal sample. After all, doesn't the digitised audio contain all the information needed, thus rendering the VCA redundant? Unfortunately (for reasons we will not discuss here), this is not the case and, if replayed directly from the ROM, the samples do *not* decay like a real cymbal. This means that something must shape the sound.

Now, if we need to shape the amplitude of the sound, it would seem straightforward to add a basic analogue contour generator and trigger this at the same time as the address counter. However, this would only work correctly if the pitch of the cymbal never changed. As you can see in Figure 2, you can tune the TR909's cymbal by altering the speed of the 30kHz oscillator that drives the counter, and this complicates matters considerably.

Think about it... If you increase the pitch of a digitised sound by increasing the clock rate, the data will be output more quickly, and you will reach the end of the samples more quickly than before. But if a conventional AR contour generator proceeds at the same rate, no matter what the speed of the digital clock, it's quite possible that the end of the samples will occur before the VCA is fully closed. If this happens, the sound will be truncated in the precise way that the sound of a real cymbal is not. So Roland made the decay rate of the AR envelope dependent upon the progress of the address counter, thus ensuring that, no matter how much you mess with the clock, the VCA always shapes the sound correctly. Clever, isn't it?

Having created the basic sound and shaped its amplitude using a VCA, you should now be able to send it to the outside world without further fuss. No? Umm... no. The missing element is the low-pass filter that lies between the VCA and the output. This is required because the signal suffers from quantisation noise and aliasing. Fortunately, the unwanted components generated by these problems lie predominantly at high frequencies, so a suitably chosen low-pass filter eliminates the worst of them without introducing unacceptable signal degradation.

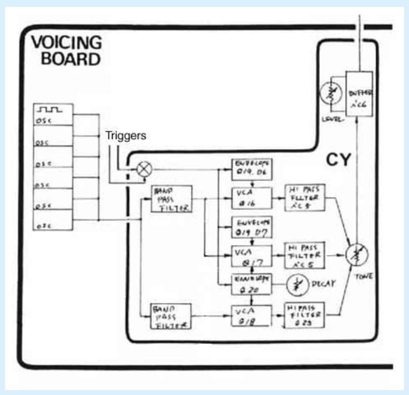


Figure 4: The TR8o8 cymbal block diagram.

And there you have it... the TR909 cymbal passes through an analogue VCA and an analogue low-pass filter. But it isn't an analogue sound. It's a set of digital samples replayed through a bit of analogue electronics.

## The TR808 Cymbal

To find a truly analogue attempt at synthesizing cymbals, we need to step back a bit further in time, and look at that other doyen of analogue drum machines, the Roland TR808.

Figure 4 (below left) shows the block diagram for the TR808 cymbal. I have reproduced this from Roland's documentation, removing all sorts of extraneous stuff to clarify things as much as possible. Figure 5 (shown on the next page) shows the same thing in a format more familiar to long-standing readers of this series. As you can see, there's nothing digital going on here.

The initial sound generator comprises six square-wave oscillators tuned enharmonically, and mixed to create a complex spectrum. If you remove all the low harmonics from the mix, this produces a moderately dense cluster of partials in the mid and high frequencies.

The mixed signal from the six oscillators is split into two bands by a pair of band-pass filters. The lower frequency band then passes through a VCA controlled by an AR contour generator. The TR808's Decay control affects the decay rate of this envelope, thus allowing you to extend or curtail the duration of the low-frequency components in the final sound.

The upper band is further split into two signal paths that pass through independent VCAs controlled by their own contour generators. The upper of the two, um, upper bands has the shortest Decay. The lower of the upper bands has a Decay that lies somewhere near the centre of the range of the low band's variable AR control. This inequality of decay times allows the TR808 to change the mix of lower-, mid- and higher-frequency components as the sound progresses.

All three bands then pass through high-pass filters to remove more yet low-frequency components, before a user-controlled mixer recombines them into a single signal (the TR808 tone control affects this mix of low, mid and high bands). Finally, an amplifier determines the loudness of the output.

## Recreating The TR808

The circuit which creates the TR808's cymbal is very elegant, and it's not trivial to recreate it ▶

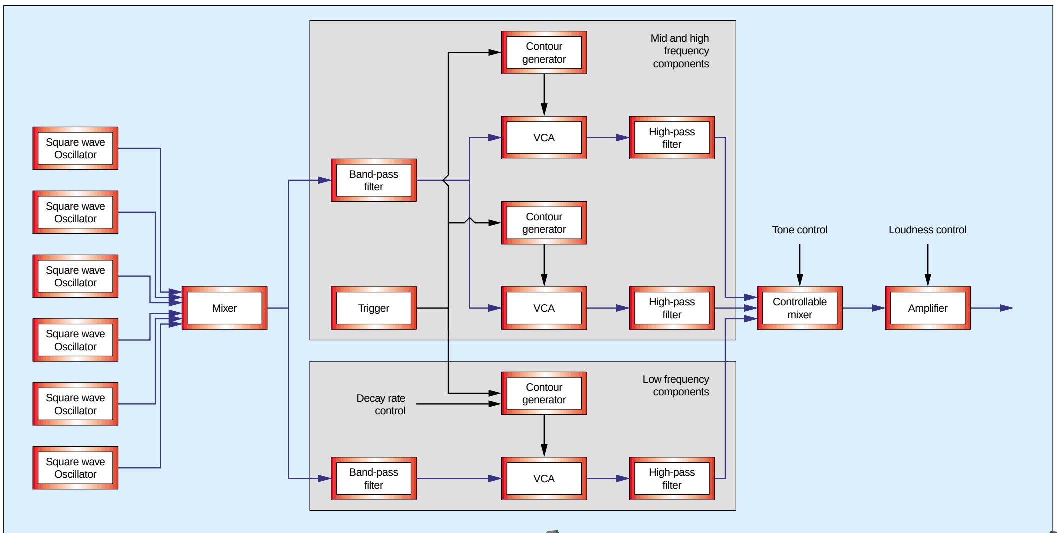


Figure 5: The TR808 cymbal in Synth Secrets format.

on a conventional analogue synthesizer. Just for fun, I've drawn Figure 6 (below), which shows a modern analogue synth — an Analogue Systems RS Integrator — configured to emulate the TR808 cymbal. That's right... this monster of a patch is *just* the cymbal in Figure 5. Note that I have not set the knobs on these modules to appropriate values... simply selected and patched the modules needed to do the job.

Adding up the cost of the modules, cases and PSUs in Figure 6, I reckon that we're looking at a collection of synth modules worth around £1500. Of course, unlike the TR808 cymbal circuit, this Integrator could produce a zillion and one other sounds, and with far higher fidelity than any analogue drum machine. Nonetheless, the diagram shows us that Roland's engineers were very smart cookies indeed.

Comparing Figures 1 and 5, we can see that they share many concepts. OK, the Roland uses multiple oscillators whereas the patch I designed used FM to generate a complex spectrum, but both models split the signal into high and low frequency bands and shape them to imitate the response of a real cymbal. If there is one area in which the TR808 falls short, it is in its lack of dynamic filtering. Last month's analysis showed that this was an important element in the sound, but in the early '80s it would probably have been too complex and too expensive to incorporate it into the TR808.

Korg's MS20.

## Synthesizing The Cymbal Sound: Part 1

Rather than patch the unrealistic monster in Figure 6, I'm now going to try — and, as you will see, fail — to create an acceptable cymbal sound on one of the common analogue monosynths that we've been using in recent parts of this series. The ARP Axxe and Roland SH101 are no good for this, because they are single-oscillator instruments with a single filter and single signal path. The Minimoog is also a non-starter. Only the Korg MS20 (shown above) approaches the complexity required. With its two signal paths and four filters, we might just about be able to patch a passable sound on it.

Let's start with the oscillators and, in particular, the primitive ring modulator on VCO2. This uses a switching circuit to modulate the pulse wave output from VCO1 (whatever the knob's setting) with the pulse wave generated by DCO2. This doesn't produce the complex cluster of partials generated by FM, and it's nowhere near as flexible as I would like, but it's the best that the MS20 can offer, so I shall attempt to use it to create the signal components needed. Sadly, a ring modulator is not best suited to this task and, although it produces four partials for each pair of VCO1 and VCO2 harmonics, the resulting output is still not sufficiently complex to resemble the metallic 'ting' we require.

I shall attempt to insert more enharmonic elements into the sound by adding the unmodulated output from VCO1, adjusted to be as 'out of tune' as possible when compared

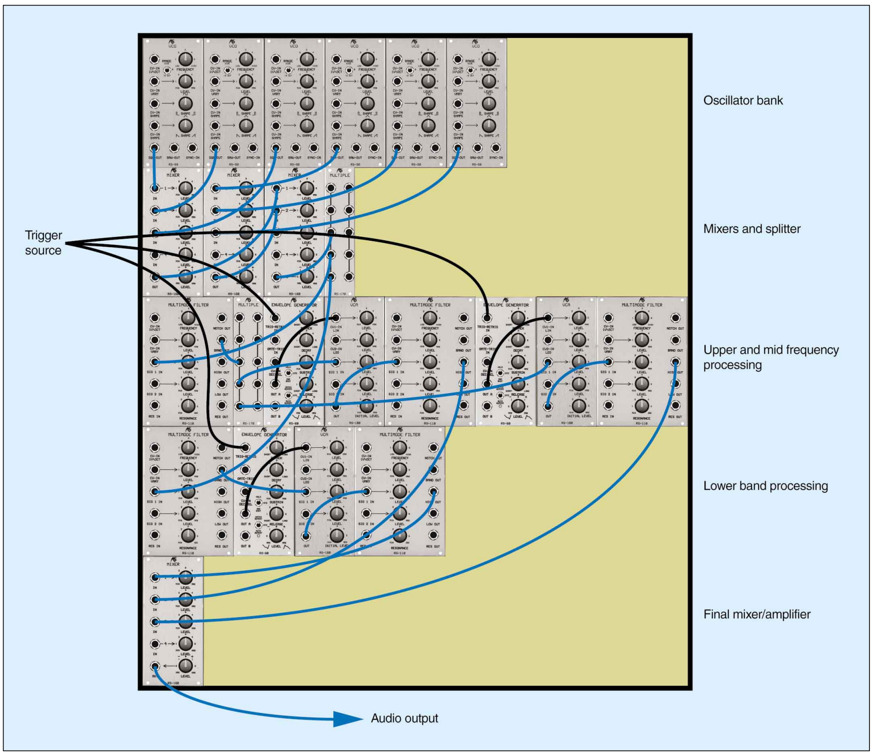


Figure 6: Recreating the TR808 cymbal on a modular synthesizer.

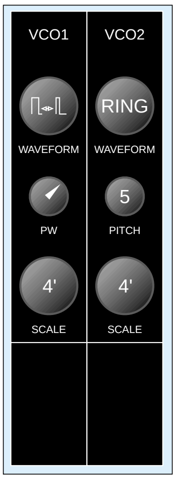


Figure 7: The MS20 oscillators producing an enharmonic timbre.

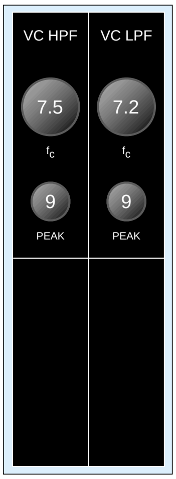


Figure 8: Producing narrow bands of highly emphasised frequencies.

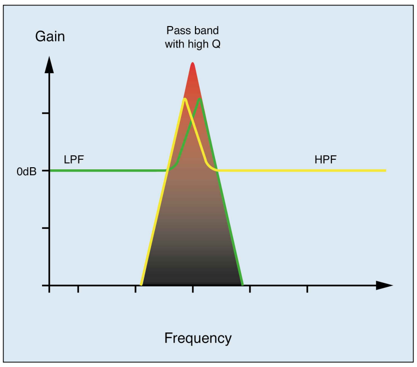


Figure 9: The response of the filter settings in Figure 8.

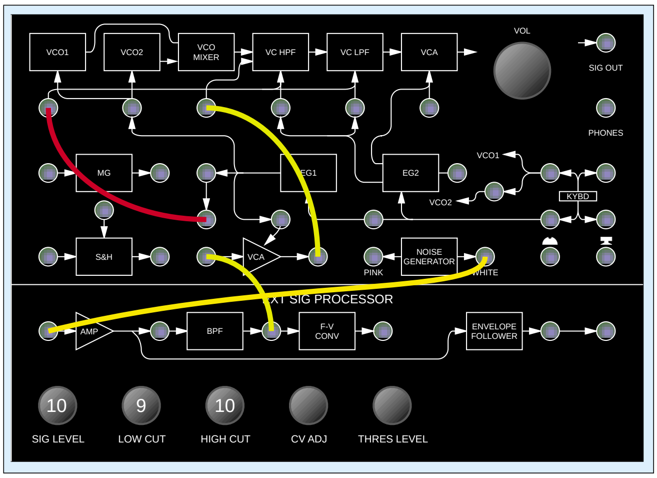


Figure 10: Using the patchbay.

to VCO2 (see Figure 7, above). But there's a problem here... the tuning of VCO1 affects the timbre produced by VCO2's ring modulator, so I shall have to find settings that satisfy both requirements. The truth is, I'm going to fail to create anything as good as the TR808, but we'll press on, and see if we can use the rest of the MS20 to improve matters.

I'll remove the low-frequency components generated by the oscillators by opening the low-pass filter and then setting the high-pass filter so that only a narrow band of frequencies survives filtering. In addition, I will emphasise the resulting band by setting the 'Peak' (Korg's term for filter resonance) to a high value for both filters, so that they are on the edge of self-oscillation (see Figures 8 and 9, left).

The sound I've produced still bears no resemblance to that of a cymbal, and the major reason for this lies in the paucity of signal components. Unfortunately, the MS20 is incapable of generating any more partials, so we must now apply a little lateral thinking if we are going to add more 'body' to the sound.

There is one way to do this, although I had hoped to avoid it because, as far as synthesis goes, it lands us back in the 1960s. But there seems to be no choice; the only place to obtain more body is from the MS20's noise generator. So now we must turn our attention to the yellow (audio signal) cables that I have inserted into the MS20's patch bay (see Figure 10, above).

As you can see, I've taken the 'white' output from the noise generator and directed this to the signal input of the External Signal Processor (or ESP), turning the input amplifier gain to maximum so that it distorts, thus roughening the sound a little. I have then adjusted the band-pass filter in the ESP to ensure that only a narrow band of high-frequency noise emerges, and directed this to the patchable VCA, and then on to the signal input that lies before the filters in the main signal path.

If you don't insert a patch cable into the patchable VCA's Gain control input, it is controlled by Envelope Generator 1, which produces a simple ASR contour. So, to shape the amplitude of this part of the sound, I must tap the keys when playing, not hold them down for an extended time. This is an acceptable compromise, and the settings (Attack to '0', and Release to '8'), as shown in Figure 11 (below), ensure that the noise signal is passed with instant Attack, and that its amplitude decays as the sound progresses.

While I have you looking at the patchbay, you should notice that there is a red (control signal) patch lead running from the Inverted output of EG1 to the 'Total' modulation input. To understand what this is doing, you must look at the modulation controls in Figure 11. You can see here that the 'MG/T.Ext' parameter for the high-pass filter is set to '7'. This means that the inverted Attack/Sustain/Release contour is affecting the cutoff ▶

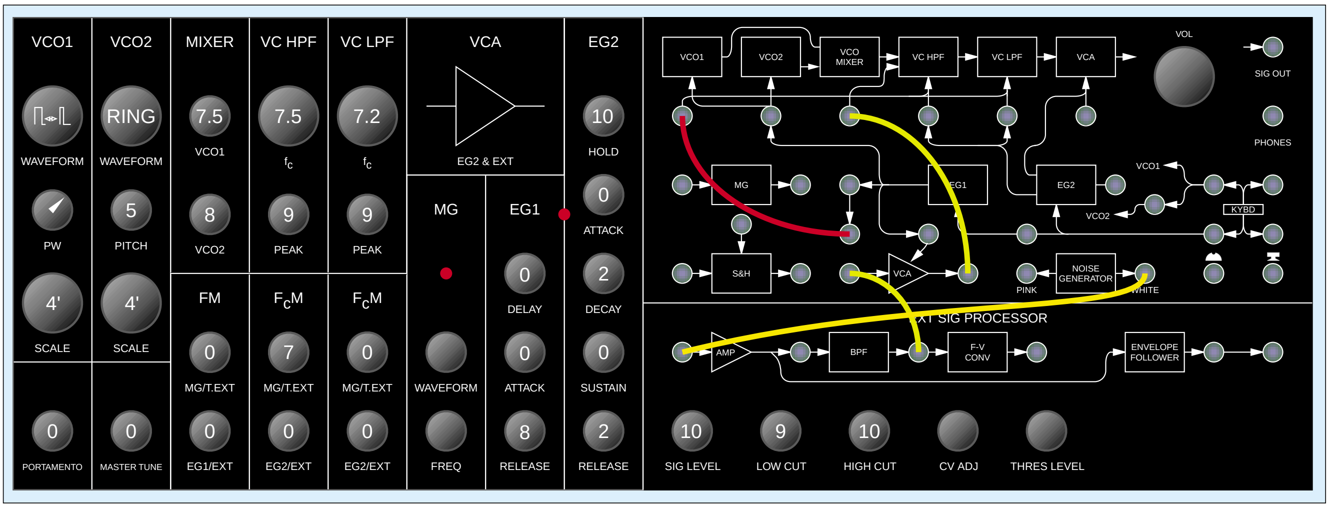


Figure 11: The MS20 cymbal patch.

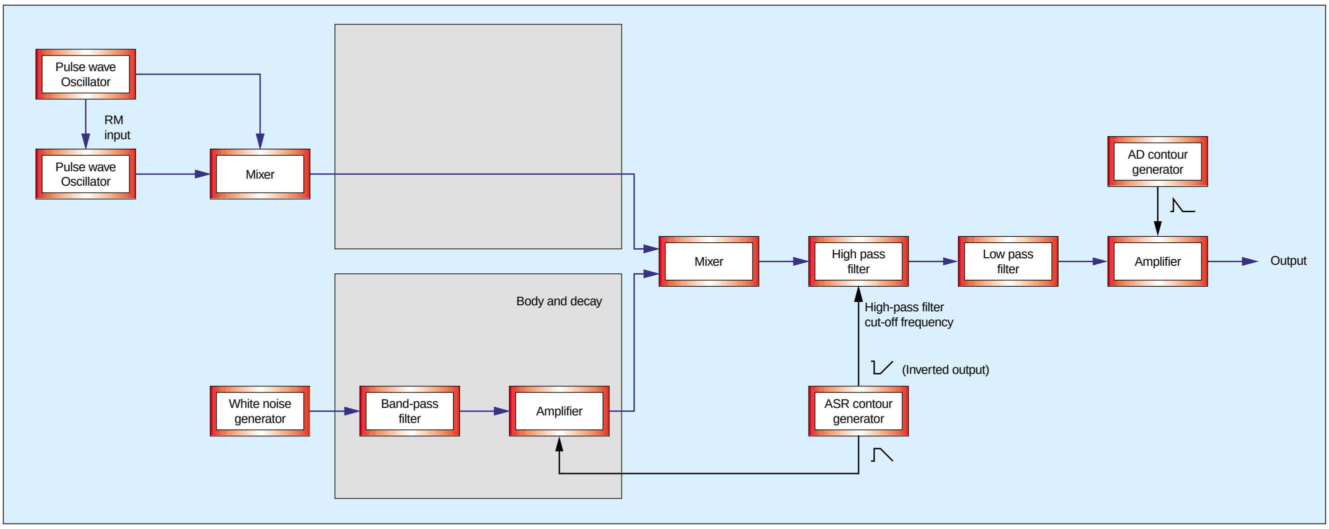


**Figure 12:** What's happening in the patch in Figure 11.

frequency of the high-pass filter, increasing the lowest pitch of the spectral content of the sound as it progresses.

Finally, I will shape the amplitude of the composite sound using Envelope Generator 2, giving the sound an instant Attack, and consistent Decay and Release of '2', thus ensuring that the envelope is the same whether I release the key or not. These settings also explain why the Sustain stage of EG1 is not a problem... Even if I hold a key for too long, the VCA controlled by EG2 will curtail the sound. What EG1 is doing, therefore, is changing the relative mix of the noise and the spectral components in the signal. It's a crude attempt to imitate one of the features of the multiple signal paths in Figure 5, but it's the best that the MS20 can offer.

If you recreate the sound in Figure 11, you will find that it is incredibly sensitive to tiny changes in the settings of the oscillators and main filters. You'll also find that, at best, it has some of the characteristics of a cymbal without ever sounding anything like a cymbal. This is disappointing, but to be expected. After all, as far as I remember, none of the patch charts supplied with analogue synths contained a cymbal patch, so you can be fairly sure that no affordable synth of this type was ever particularly well-suited to reproducing this kind of sound.

## What Went Wrong?

Figure 12 (above) shows the signal path for the patch in Figure 11, and when we compare this with the model in Figure 1, we can see why the MS20 cannot do the job we have asked of it. As already noted, the initial sound lacks a metallic 'ring', and much of the body of the sound comprises noise rather than discrete signal components. In addition, the 'impact' synthesis is missing, and the contour in the body affects only the VCA, not the band-pass filter. Ultimately, these and other limitations have proved to be too great, and so the MS20 proves inadequate for the task.

You might ask, therefore, whether this invalidates the patch in Figure 11. I don't think that it does. It is an extremely aggressive sound that would punctuate any rhythm track. Just don't believe that it sounds like a real cymbal, because it doesn't.

## Synthesizing The Cymbal Sound: Part 2

Let's now returns to the Nord Modular patch that I developed last month. This made a much better job of creating the semblance of a cymbal than does the MS20, but still leaves room for a single, enormous improvement...

I suspect that Roland adopted its approach of using multiple oscillators because it's cheap and easy to build coarse square-wave oscillators. However, a rival company, Korg, discovered that using multiple pairs of *modulated* square-wave oscillators creates a much more authentic metallic timbre. So, while Roland seemed happy with the TR808, Korg was manufacturing drum machines such as the Rhythm 55, which offered cymbal sounds that were an order of magnitude more complex — and more realistic — than those produced by the Roland. And therein lies the secret to affordably synthesizing all manner of metallic percussion.

Given the nature of the Nord Modular, it's simple to enhance last month's patch by adding more pairs of modulated oscillators, detuning them in ways that sound appropriate, and then mixing them to a single signal (see Figure 13 on the next page).

The results are stunning, capturing the very essence of metalwork. Playing with the oscillator waveforms and frequencies, together with the contours controlling the filter and amplifier in the rest of the patch (shown in Figure 14 on the next page) produces large cymbals, small cymbals, Eastern cymbals, rides, crashes, hi-hats, the finger cymbals in tambourines... The possibilities (which we will explore in more detail next month) are enormous.

## Epilogue

Let's finish by re-evaluating what we have learned about cymbals. Last month, and the month prior to that, I discussed the nature of the cymbal and conducted an empirical analysis, using this to create a recognisable cymbal sound with just a couple of digital oscillators, filters and amplifiers within a Nord Micro Modular. This month I showed that affordable analogue synths are unsuitable for synthesizing cymbals, but that — if we learn the lessons of the clever, dedicated sound generators in drum machines — we can create much better emulations using powerful modular synths or software synths.

Of course, you might assume that in these days of supercomputers, Pentium-driven software synths, and workstations containing up to a dozen DSPs, it will soon be possible to step beyond Figure 14, and model the precise sound of a cymbal without resorting to modulated oscillators or samploid synthesis. However, this belief is probably flawed, for the following reasons.

Last month, I stated that an analysis of the cymbal is far beyond the scope of these articles. What I omitted to say is that a complete analysis of the cymbal currently remains beyond the scope of *anybody*! Cast your mind back to the analogy I made two months ago, where I related the means by which a cymbal produces its sound to what happens when a stone is thrown into a pond. Now try to imagine what would happen if the ripples produced by the impact took days to dissipate, interacting and interfering with one another everywhere on the water's surface, and doing so differently at every point and at every moment in time. Now add complicating factors such as, for example, an uneven pond-bed, and make the water more viscous at some points than it is at others... A full analysis of this is all but impossible, and a numerical model of its behaviour would require almost infinite processing bandwidth. Thankfully, both the Korg Rhythm 55 and the Nord patch in Figure 14 prove that (unlike our attempts at guitar synthesis earlier in this series) we need neither a PhD in acoustics nor God's own Personal Computer to synthesize acceptable cymbal sounds. And that news, surely, will come as a relief to synthesists everywhere. SOS

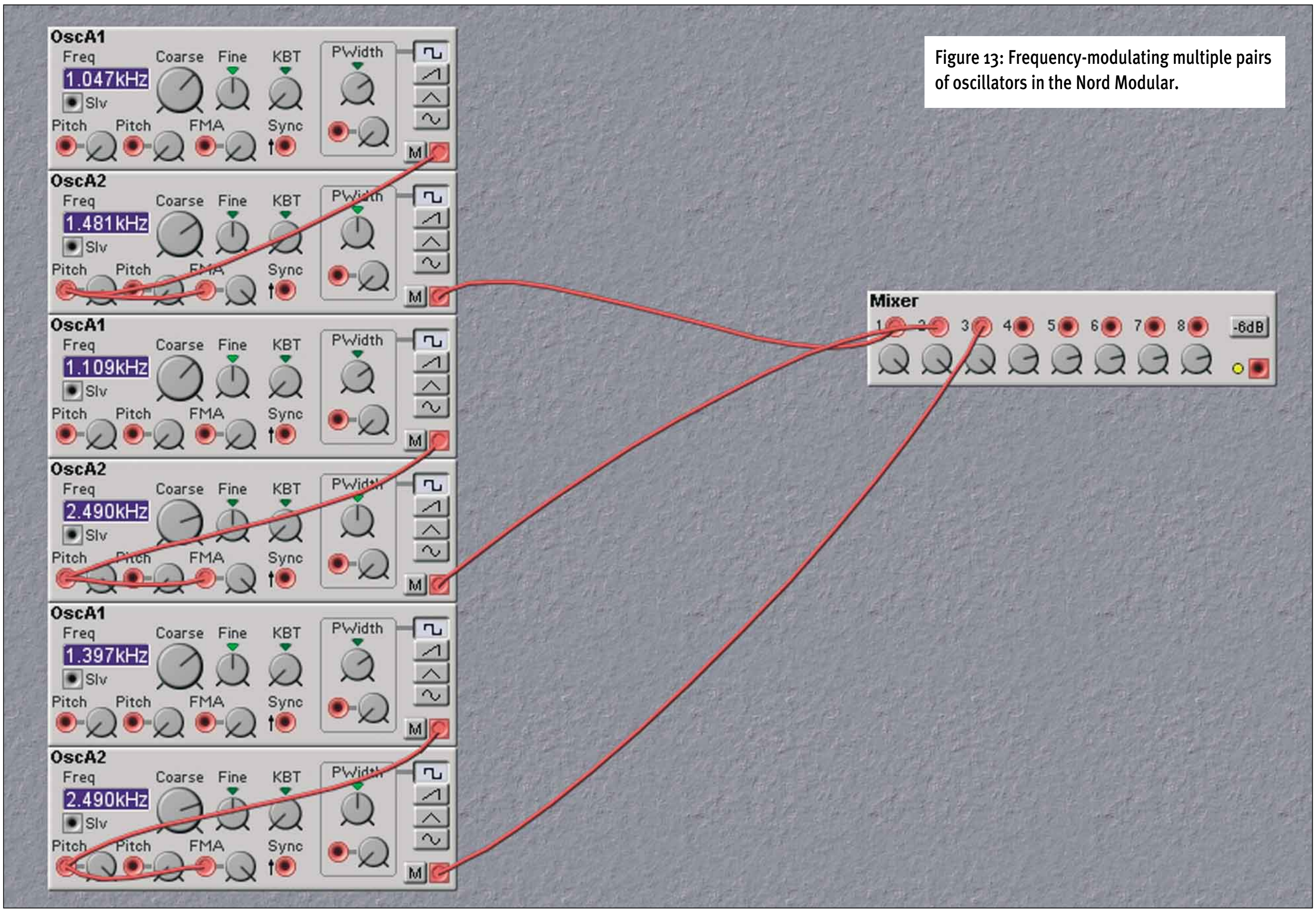

OscA1 Freq 1.047kHz
OscA2 Freq 1.481kHz
OscA1 Freq 1.109kHz
OscA2 Freq 2.490kHz
OscA1 Freq 1.397kHz
OscA2 Freq 2.490kHz
Mixer -6dB

Figure 13: Frequency-modulating multiple pairs of oscillators in the Nord Modular.

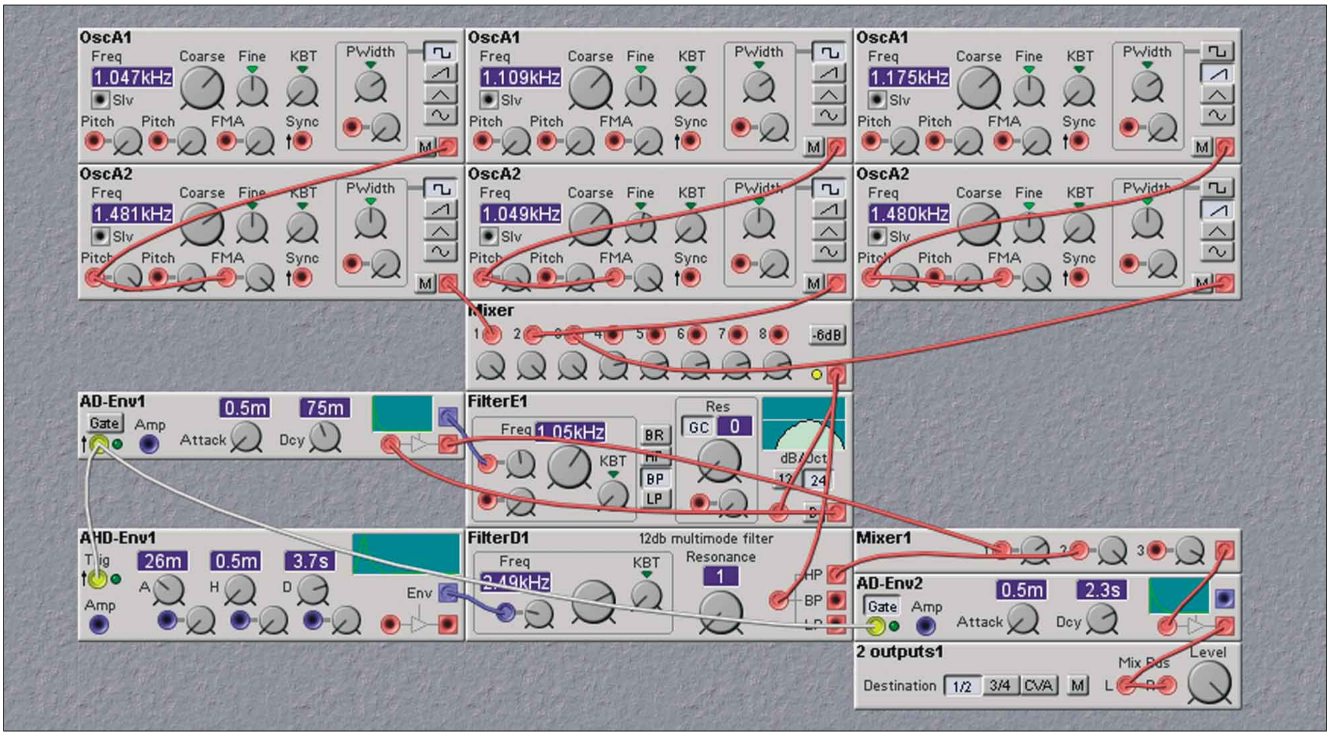

OscA1 Freq 1.047kHz
OscA1 Freq 1.109kHz
OscA1 Freq 1.175kHz
OscA2 Freq 1.481kHz
OscA2 Freq 1.049kHz
OscA2 Freq 1.480kHz
Mixer -6dB
AD-Env1 Attack 0.5m Dcy 75m
FilterE1 Freq 1.05kHz Res 0 GC dB/Oct 12 24
AHD-Env1 A 26m H 0.5m D 3.7s
FilterD1 12db multimode filter Freq 2.49kHz Resonance 1
Mixer1
AD-Env2 Attack 0.5m Dcy 2.3s
2 outputs1 Destination 1/2 3/4 CVA

Figure 14: Using multiple frequency-modulated oscillators to create a better cymbal sound.

# Synth Secrets

## Synthesizing Bells

**Having come up last month with a reasonably realistic cymbal patch, it's time to take the principles of synthesizing metallic percussion one stage further, and produce bell sounds. But there's more to this than you might think...**

Gordon Reid

Though you may not have been aware of it, for the past three months we've been investigating the sound-generation mechanism of a particular class of *idiophones*, the cymbals. Cymbals, hi-hats, and bells are all idiophones, but the family also includes instruments such as marimbas and xylophones. You might think that these share few characteristics with cymbals, but they are all rigid objects that require no tensioning mechanism in order to vibrate and produce a sound, in contrast to the *membranophones* discussed in past Synth Secrets (such as the snare and bass drums).

### Taking It Further

Having found that we can recreate the cymbal sound with reasonable realism, you might be forgiven for thinking that we can use our cymbal patch to synthesize other metallic percussion instruments. And, in part, you would be right. To demonstrate this, let's consider the cymbal patch in Figure 1 (right), with which I concluded last month's Synth Secrets. The key to this patch is the use of the six oscillators arranged as three frequency-modulated pairs. These produce a dense fog of enharmonic partials that, without need for any further treatment, sound inherently 'metallic'.

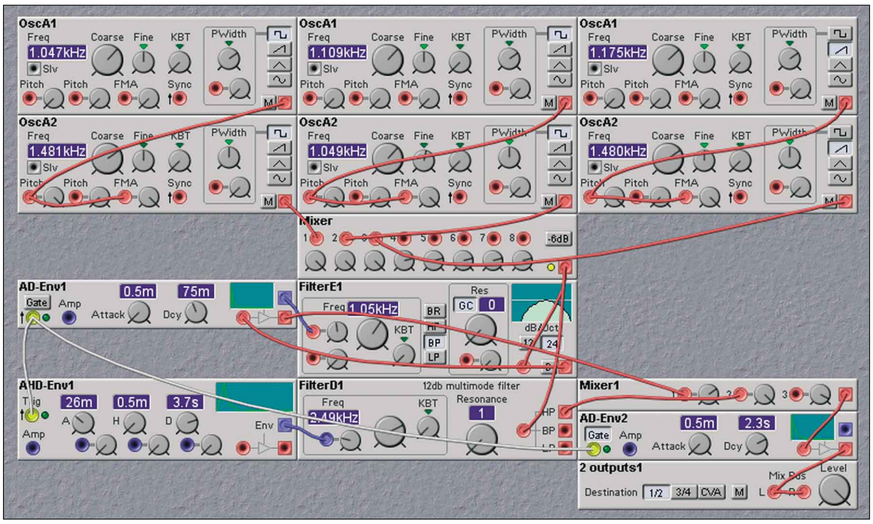
It therefore seems reasonable to suppose that we can adjust the parameters in Figure 1 to emulate a range of percussion instruments related to cymbals. And, as suppositions go, this is not a bad one. For example, shortening the envelope times allows you to synthesize very acceptable hi-hats. Making the envelopes briefer still produces excellent imitations of the stick hitting the hat, and careful adjustment of the filter frequencies, envelope times and mixer settings (the last of which controls the relative loudness of the stick impact and the body of the sound) creates very realistic effects. You can even emulate the opening and closing of the hats by adding modules such as the Control Sequencer at the bottom left of Figure 2. This modifies the decay rate of envelope AHD-Env2, creating subtle changes in the sound and ensuring that the patch sounds more interesting (and more realistic) than the static samples found in most drum machines and samploid synths.

Unfortunately, if we follow this line of investigation any further, we'll find that we are doing nothing more challenging than synthesizing different sizes and thicknesses of cymbals and related instruments. In other words, a flat plate bashed into the shape and size of a cymbal is much like a flat plate bashed into the shape and size of a hi-hat, and so on. So, to move our understanding of idiophones forward, we must consider the case where the instrument exhibits different properties from the above.

A few months ago, we did a similar thing by taking a membranophone and adding a snare to its carry head. This simple alteration changed the physics significantly, creating two very different percussion instruments; the bass and snare drum. This month, we're going to do something equally simple; change the shape of the instrument. We're going to talk about bells.

### From Cymbals To Bells

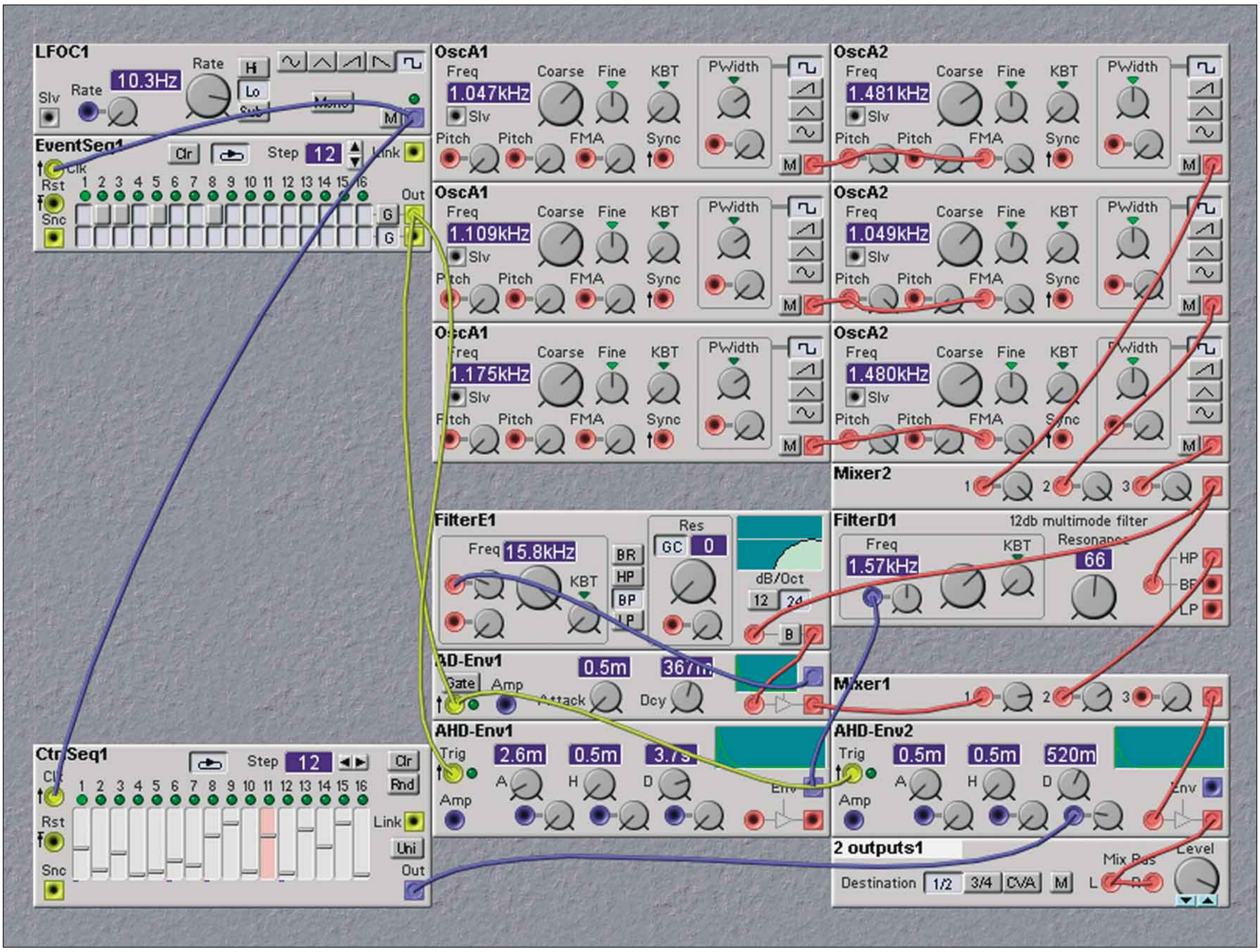
At first sight, it might seem simple to turn a cymbal into a bell... you just push down the edges and, notwithstanding a lot of crinkling, you'll eventually create a bell shape (see Figure 3, below left) You might therefore expect that, if you hit this with the same sticks and in the same way as you did



OscA1 Freq 1.047kHz
OscA1 Freq 1.109kHz
OscA1 Freq 1.175kHz
OscA2 Freq 1.481kHz
OscA2 Freq 1.049kHz
OscA2 Freq 1.480kHz
Mixer -6dB
AD-Env1 0.5m 75m
FilterE1 Freq 1.05kHz Res GC 0
AHD-Env1 26m 0.5m 3.7s
FilterD1 12db multimode filter Freq 2.49kHz Resonance 1
AD-Env2 0.5m 2.3s
2 outputs1 Destination 1/2

Figure 1: Synthesizing a cymbal using the Nord Modular.

Figure 2: A hi-hat patch for the Nord Modular.



LFOC1 Rate 10.3Hz
EventSeq1 Step 12
OscA1 Freq 1.047kHz
OscA2 Freq 1.481kHz
OscA1 Freq 1.109kHz
OscA2 Freq 1.049kHz
OscA1 Freq 1.175kHz
OscA2 Freq 1.480kHz
Mixer2
FilterE1 Freq 15.8kHz Res GC 0 dB/Oct 12 24
FilterD1 12db multimode filter Freq 1.57kHz Resonance 66
AD-Env1 0.5m 367m
Mixer1
AHD-Env1 2.6m 0.5m 3.7s
AHD-Env2 0.5m 0.5m 520m
CtrlSeq1 Step 12
2 outputs1 Destination 1/2 3/4 CVA

before, it would sound similar to the cymbal. However, as experience tells us, it does not.

This is because bells are much more complex than Figure 3 would suggest. Sure, there are some that are, in essence, bent sheets, lacking internal structure and perhaps even displaying a weld along one edge. But, due to the change in geometry, hitting one of these produces a sound very different from that of a cymbal. Some bells are not unlike cylindrical shells with end-caps, and these too sound very different from other idiophones. Then there are sleigh-bells, wooden bells shaped like seed pods... and many others, all of which produce distinctive sounds easily distinguished from one another as well as from cymbals and hi-hats.

The types of bell with which we are most acquainted are church bells and hand-bells. Unlike cymbals and hi-hats, these are not bent sheets of metal, and they do not exhibit the acoustic properties of plates. These bells are complex shapes cast from molten metal and lathed internally, which modifies the sound in various desirable ways, as we shall see. The consequence of this is that, while we use the same mathematical tools to analyse church bells as other metal idiophones, and while the shapes of the vibrations are analogous to those of plates, church bells have quite a different set of acoustic properties to anything we have previously discussed — which is, of course, why they sound so different.

There are many parts to a church bell. The top is the *crown*, and this is followed by the *shoulder*, the *waist*, the *bow*, and the *lip*. The opening at the bottom is the *mouth*. Clearly, bells are very human lumps of metal (see Figure 4, below).

There are two common ways used to energise a bell like this. The first is the method found in church and schoolyard bells. These have internal metal *clappers* that strike the bow when the bell is rocked or shaken. The second is to use an external hammer or a clapper connected to a mechanical lever. When a number of such bells are placed together and tuned to a chromatic scale, and the levers are arranged in a conventional keyboard layout, the result

▶

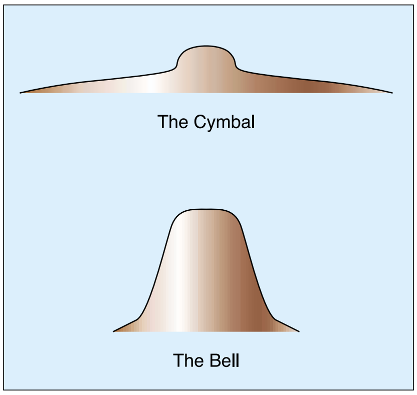


Figure 3: Simplified representations of the cymbal and the bell.

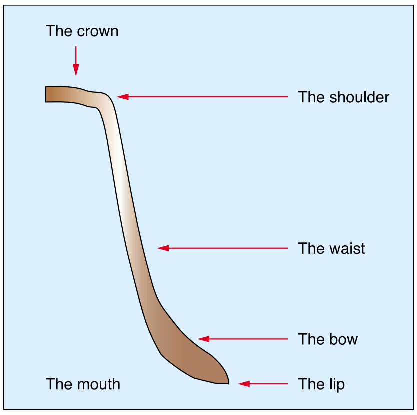


Figure 4: The parts of the bell.

▶ is the world's heaviest musical instrument, the *carillon*.

Hang on a second... tuned to a chromatic scale? If we can tune bells to have pitches that we can play from a keyboard, they must be *very* different from cymbals, which, as we know from last month, produce highly complex, atonal timbres.

## The Modes Of A Bell

Like any other musical instrument, a bell's timbre is determined by the pitches and amplitudes of the partials that constitute its sound. And, as we have discussed before, these partials are related to modes of vibration. In a bell, the modes are described by the number of circular nodes around the body of the bell, and the number of radial nodes that we can trace from the lip on one side, over the crown, and down to the lip on the other side. I have shown one example of a bell mode in Figure 5. You can see clearly the two radial and two circumferential nodes that define the 2,2 mode shown.

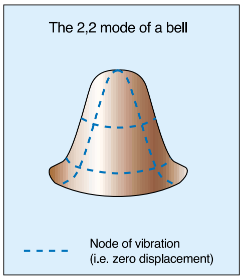


Figure 5: The 2,2 mode of a church bell.

However, I can't now follow previous Synth Secrets practice and draw a selection of low-order modes for the bell — it would soon drive me (and probably you) crazy. This is because, unlike the (essentially) two-dimensional heads of the bass drum and snare drum, and the cymbals discussed over the past few months, bells are three-dimensional structures. To give you some idea how this complicates matters, Figures 6 and 7 show the mouth and a vertical section of the bell oscillating as shown in Figure 5.

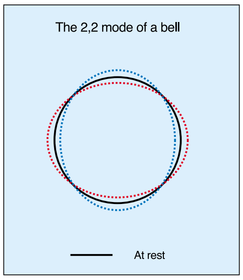


Figure 6: Looking up into the mouth of the bell with a 2,n mode excited.

In Figure 6, the black circle represents the shape of the mouth when the bell is at rest. The blue line shows the distortion of the mouth at some arbitrary moment when the bell is ringing with 2,n motion (where 'n' is any whole number), while the red line shows the distortion due to that mode half a cycle later.

Figure 7 shows a cross-section of one half of the bell when ringing with an m,2 motion (where 'm' is any whole number).

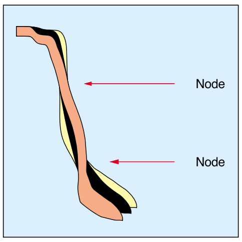
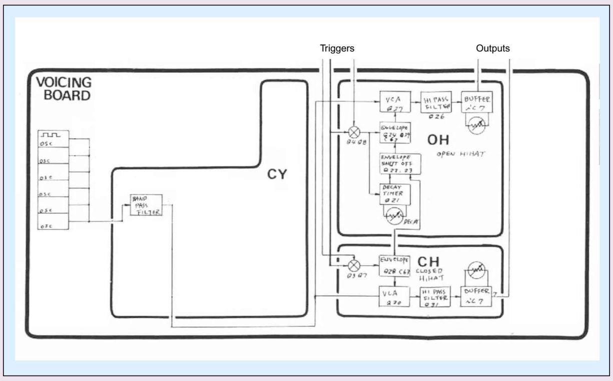


Figure 7: A cross-section of half a bell, showing the motion when an m,2 mode is excited.

## The TR808 Hi-hat

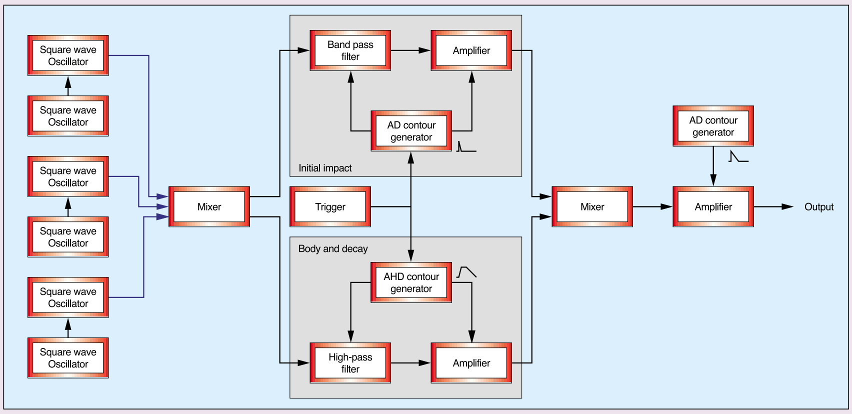
**As in previous parts of this series, I thought that it would be useful to compare the hi-hat patch in Figure 2 with the hi-hat circuits in the Roland TR808 drum machine (see diagram, right). Although this diagram and Figure 2 look different, they are in many ways structurally quite similar to one another — exactly as you might expect, since Roland's engineers and I were trying to create the same sound. Furthermore, the TR808's six square-wave oscillators were one of the inspirations for the six pulse-wave oscillators in my cymbal patch, from which Figure 2 is derived.**



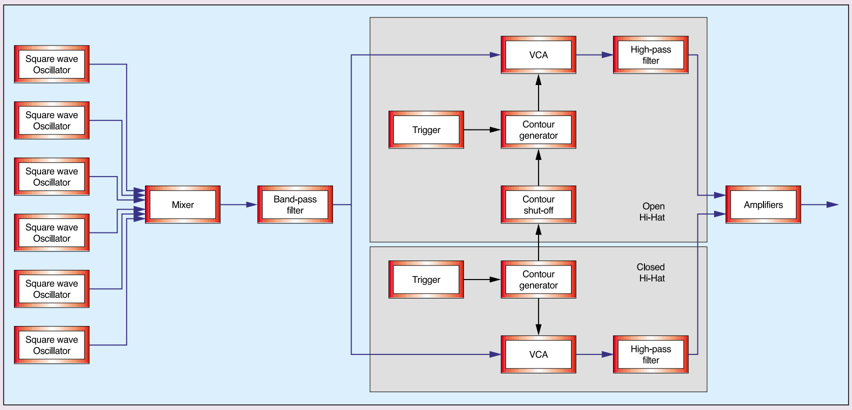
The hi-hat generators in the Roland TR808.

**To demonstrate the similarity between the two architectures, I have redrawn both in standard Synth Secrets-style format. The second diagram below shows the signal flow within the Nord patch, and the final diagram in this box does the same for the TR808.**

**Sure, there are differences between the two. Nevertheless, the essential elements are the same: multiple oscillators are fed through a filter that removes the low frequencies, and through a VCA controlled by a simple contour generator. The only extra element in the TR808 is a 'shut off' circuit that curtails the decay of the Open Hi-Hat contour generator when a Closed Hi-Hat is detected. This is not necessary in the Nord patch, because the same signal path produces both voices.**



The sound generator from Figure 2 redrawn.



The TR808 hi-hat sound generator redrawn.

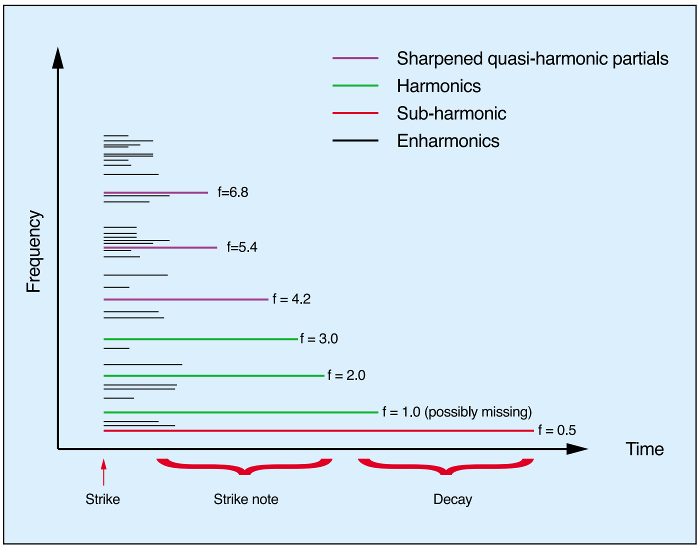


Figure 8: Representing the three phases of the bell's sound.

Again, the black section represents the shape of the bell when the bell is at rest. It should now be clear that the pink section shows the distortion of the bell at some arbitrary moment, while the yellow section shows the distortion due to that mode half a cycle later.

Perhaps you can picture how these motions coexist in the 2,2 mode. Now try to imagine how this relatively simple mode co-exists and interacts with other modes. Actually, it's not as bad as you might think. Just try to imagine the bell as a flat plate — ie. reverse the transformation in Figure 3 — and you will see that the motions relative to its surface are analogous to those of the cymbal modes in Synth Secrets, April 2002.

## The Bell's Sound

Next, we'll consider the surprising property shared by most traditional bells, but not by cymbals and their brethren. Many centuries ago, the people who cast metal bells discovered that they could make them sound more 'musical' by shaping the inside carefully. They probably didn't know it at the time, but they were modifying the positions of the nodes, and thus the frequencies at which they vibrated, until they generated a quasi-harmonic series. This is why you can play tunes on bells. Nonetheless, bells are not the same as simple harmonic oscillators. If they were, they might sound like hammered strings. So what gives bells their identifiable timbre?

A bell exhibits one type of behaviour at the start of a note, and different behaviour as the note decays. Experiments show that there are, in fact, three distinct phases to the sound. The first is the *strike* — the sound of one large lump of metal hitting another. As you would expect, this is enharmonic, and it dies away quickly. The second phase is the strike note, and this is dominated by a handful of strong, low harmonics. Finally, the note's lingering energy is radiated by a sub-harmonic an octave below the fundamental.

The strike note is particularly interesting, because the perceived pitch is not necessarily the pitch of the lowest energetic partial. Do you remember the organ builder's trick I mentioned when I discussed frequency-shifters back in January's instalment of this series? I described how, if the partials of a sound lie in an harmonic pattern based on a fundamental frequency that *isn't* present in the signal, the human brain inserts the missing pitch, and you 'hear' the fundamental, even if it's not actually there (see www.sound-on-sound.com/sos/jan02/articles/synthsecrets0102.asp to read the whole article). The strike note of a well-tuned bell does the same thing. The dominant partials can be tuned to produce frequencies in the ratios 2:3:4, so that the listener hears the implied pitch of '1'. For example, if the dominant partials vibrate with frequencies 100Hz, 150Hz and 200Hz, you will 'hear' a fundamental of 50Hz.

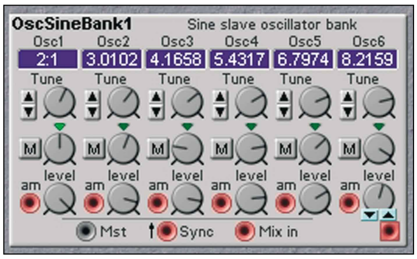
Putting all of this knowledge together, we can create Figure 8 (above). This is only a graphic representation, but it shows the three phases of the sound in an easily understood form. As you can see, an initial burst of enharmonic partials is followed by an extended period in which the low harmonics (some of which grow progressively sharper with increasing frequency) determine the sound. Below these, there lies the subharmonic that dominates the sound in its final moments.

Of course, the sound of a real bell is much more complex than this. I have ignored numerous factors such as the changes that occur when the bell is struck with clappers of different materials or at different speeds, as well as those that occur when bells are cast of different alloys, or of different sizes and relative dimensions. Fortunately, we can ignore all of these here, although we must include one additional factor if we are to synthesize realistic bell sounds. Like most adults in the bath, bells *warble*... rather than producing a steady 'boing', many bells make a noise closer to 'boii-yoy-yoy-yoiinnnggg' because their modes can be almost *degenerate*. This means that bells produce two partials of almost (but not quite) identical frequencies, and these interfere (or 'beat') in the same way as do two synth oscillators of similar frequency.

## Synthesis

Given all the above, we now have sufficient information to synthesize a bell. We'll start with the strike note. Since we require a small number of partials with 'stretched' harmonics, we can't use conventional analogue oscillators. As I've mentioned many times before, the partials of the waveforms generated by analogue synths lie in a perfect harmonic series (ie. 1:2:3:4:5:6... and so on) and will not sound correct here. Bells are much better synthesized using additive synthesis.

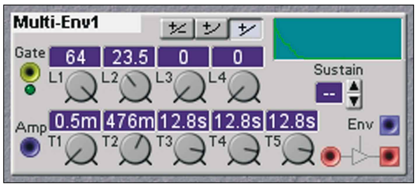
Clavia's Nord Modular has a module called 'OscSineBank', which is perfect for this purpose. Generating six sine waves of freely tuneable frequencies and amplitudes, it is ideal for creating the stretched harmonic spectrum that is the basis of the bell's sound (see Figure 9 above).


OscSineBank1 Sine slave oscillator bank

| Osc1 | Osc2 | Osc3 | Osc4 | Osc5 | Osc6 |
|---|---|---|---|---|---|
| 2:1 | 3.0102 | 4.1658 | 5.4317 | 6.7974 | 8.2159 |

Figure 9: Generating a stretched spectrum comprising six partials.

Having set this up, we could shape the amplitude of the sound using a multi-stage envelope generator/amplifier. This would allow us to create a dual decay with a rapid initial peak followed by a slower decay/release (see Figure 10, below). However, I'm going to direct the output from the oscillator bank to two envelope generators, the amplifier of one of which is being modulated by an LFO. This allows me to create the warble. Alright, it affects the


Multi-Env1

| | L1 | L2 | L3 | L4 |
|---|---|---|---|---|
| Gate | 64 | 23.5 | 0 | 0 |

| | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|
| Amp | 0.5m | 476m | 12.8s | 12.8s | 12.8s |

Sustain --

Figure 10: A dual-stage decay for the bell sound.

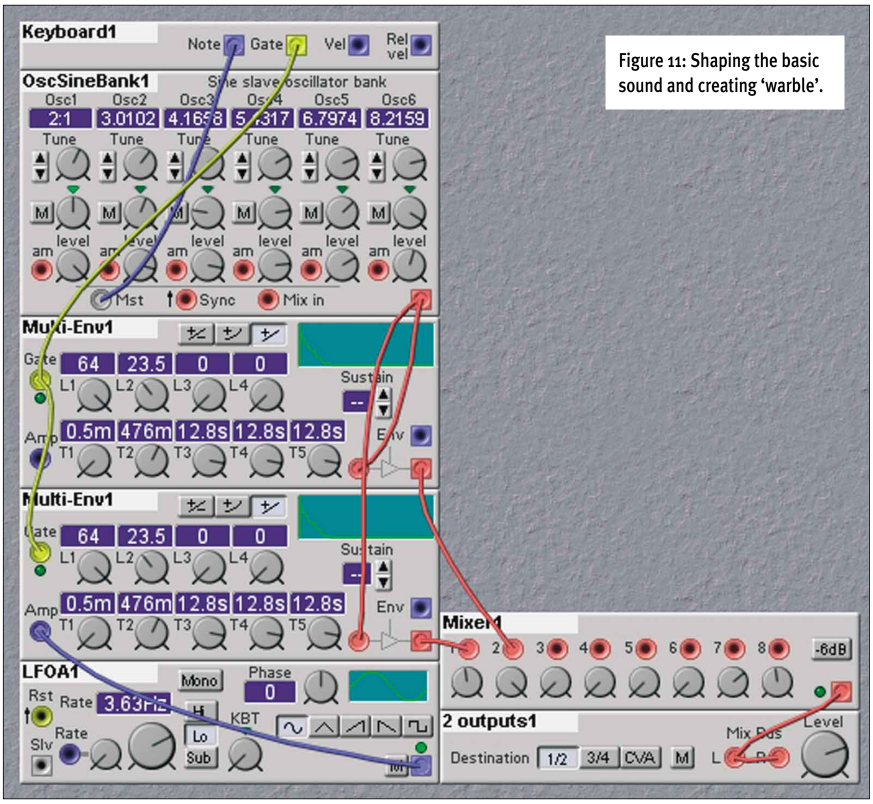


Figure 11: Shaping the basic sound and creating 'warble'.

amplitude envelope of the whole sound rather than individual harmonics, but the effect is subjectively good (see Figure 11, above).

We must now add sound generators for the initial impact of the clapper, and for the subharmonic that lingers at the tail of the sound. I have generated the first of these with two-operator sine-wave FM synthesis, the output from which is passed through another envelope generator/amplifier with shorter decay/release times than the ones in Figure 11. The resulting sound is not particularly complex, but it is atonal, and shorter-lived than the 'body' of the sound (see Figure 12, right). In isolation, this part of the patch does not sound great, but together with the rest of the sound, I find that it adds a subtle something that I like.

The subharmonic hum is generated using two further sine-wave oscillators set approximately an octave apart, but detuned just enough to create a slight beat in the sound. These are passed through another multi-stage envelope generator/amplifier, but this time with a slower attack and more extended tail (see Figure 13, below right).

Putting everything together, we now have a patch that combines a set of short (-ish) atonal components, an extended quasi-harmonic series that warbles, and a long tail comprising a couple of slowly beating low-pitched subharmonics (see Figure 14, below). Theory tells us that this should sound much like a bell, and guess what... it does!

## Epilogue

If you've rushed off to create this patch, and found that it sounds incredibly unrealistic across most of the keyboard, please don't write to *Sound On Sound* to tell me — I know. There are several reasons for this, the most important of which is that I designed the patch for use on middle 'C'. Stick to this, and you can hear how it creates the various components of the bell's unique sound (it also works well on 'C#', 'D' and 'D#'.) Sure, it could all be done in other

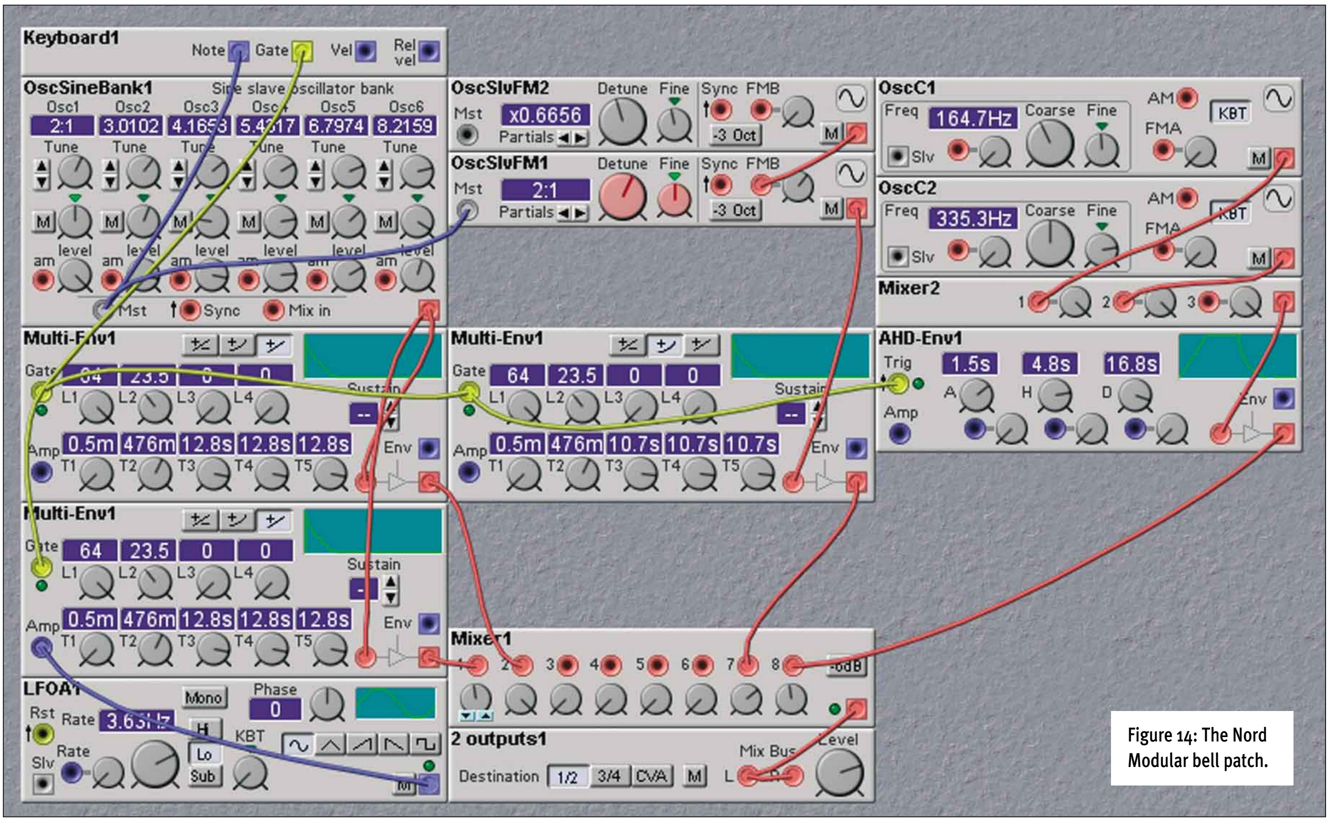
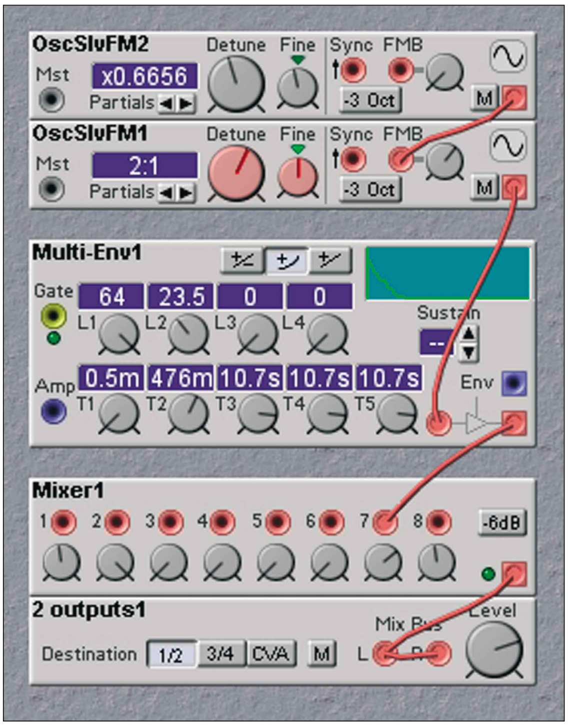


Figure 14: The Nord Modular bell patch.



OscSlvFM2 Mst x0.6656 -3 Oct

OscSlvFM1 Mst 2:1 -3 Oct

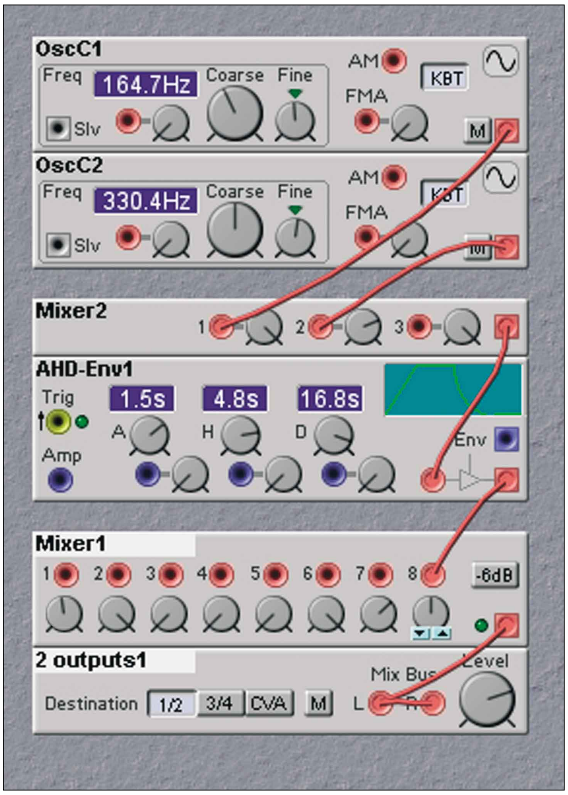
Multi-Env1 Gate 64 23.5 0 0 Sustain -- Amp 0.5m 476m 10.7s 10.7s 10.7s

Mixer1 -6dB

2 outputs1 Destination 1/2 3/4 CVA M

Figure 12: An atonal patch to emphasise the clapper sound.

ways, but that's not the point. I designed this patch from first principles in about five minutes, and it sounds very much as I intended (ie. like a bell). Given that musical inspiration can evaporate more quickly than methanol, it's important to be able to create sounds quickly and efficiently when working on a track. A firm understanding of the physics of the bell enabled me to do this. See... all this science and analysis really works! SOS



OscC1 Freq 164.7Hz KBT

OscC2 Freq 330.4Hz KBT

Mixer2

AHD-Env1 Trig 1.5s 4.8s 16.8s

Mixer1 -6dB

2 outputs1 Destination 1/2 3/4 CVA M

Figure 13: The subharmonic 'hum' at the tail of the sound.

# Synth Secrets

## Synthesizing Cowbells & Claves

**Having learned last month how to synthesize tuned bells, we turn this month, in the last of this series on the subject of percussion, to untuned bells — in the form of the humble cowbell — and claves.**

Gordon Reid

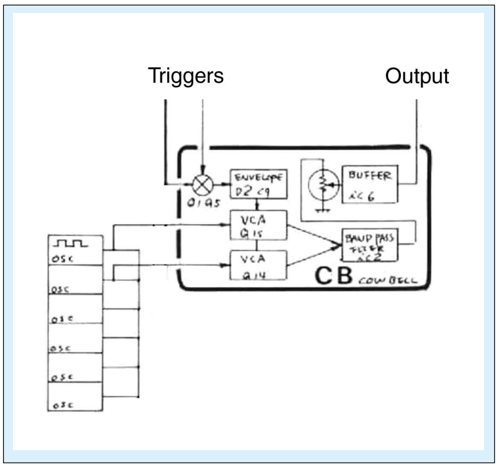


Figure 1: The Roland TR808 cowbell.

Last month, while analysing the sound of the church bell, I decided to conduct a quick literature search, and found that there are few academic texts discussing the sound generated by these instruments. On the one hand, this seems surprising, if only because bells are amongst the oldest instruments invented by human beings. On the other, it's not so surprising, because bells are very individual, and although any two examples are recognisable to our ears as belonging to the same family, they can exhibit significantly different properties from one another. No matter. We can still proceed to recreate different types of bell sounds by listening to them, and then recreating them on a suitably equipped synth. This is the *empirical* approach, and it can be just as valid as a more analytical method.

It's interesting to note that most societies have developed bells of one sort or another, and that in each case they evolved along similar lines. This is inevitable; it's inconceivable that a civilisation could cast a 200-kilo cathedral bell before hammering a small cowbell out of a simple sheet of metal, and it therefore follows that the earliest metal bells were exactly that... small, hammered from a sheet of metal, and not dissimilar to the cowbells now used by alpine herdsmen and drummers alike.

Given the significant differences between them, you won't be surprised to discover that cowbells are quite unlike the church bells and handbells that we discussed last month. These, as I discussed, are shaped to produce distinct pitches with recognisable harmonic series. In contrast, cowbells conform more closely to the concept of three-dimensional plates, something I mentioned as far back as part two of this series (see SOS June 1999, or www.sound-on-sound.com/sos/jun99/articles/synthsecrets.htm). Unfortunately, synthesizing realistic cowbell timbres has proved to be difficult using subtractive techniques in the past, and I am not aware of any programmable analogue synth that offers a convincing patch. In fact, I stumbled across my favourite cowbell patch quite by accident, as we will see towards the end of this article.

The story starts, as it has done before, with an analysis of the analogue cowbell sound on the Roland TR808.

### Cowbells On The TR808

Figure 1 (left) shows the block diagram for the cowbell sound generator in the TR808. This is a relatively simple circuit, and uses just two of the six pulse-wave oscillators that provide the basis of the machine's cymbal and hi-hat sounds. The outputs from these pass through a pair of VCAs controlled by a contour generator, and through a band-pass filter that removes the upper and lower partials. Finally, the result is then amplified before reaching the outside world. I have redrawn this in standard Synth Secrets format in Figure 2 (see below).

We should be able to recreate this sound on any synth with two oscillators and a band-pass (or dual high-pass/low-pass) filter section. But first, we can simplify the patch by eliminating the pair of VCAs to the left of the mixer, and replacing them with a single VCA after it. This is because both VCAs in the TR808 circuit are responding to a single contour generator. I suspect that this architecture was chosen in order to dispense with the mixer (if you study Figure 1, you'll see that the VCAs' outputs are hard-wired to the filter input). After making this change, we end up with the simplified block diagram that

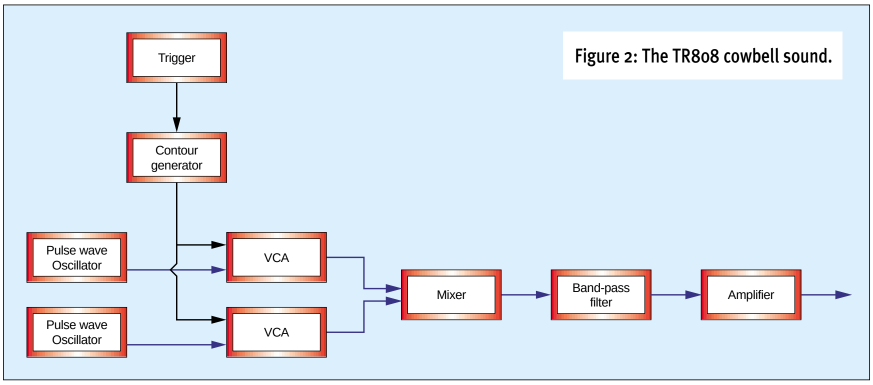


Figure 2: The TR808 cowbell sound.

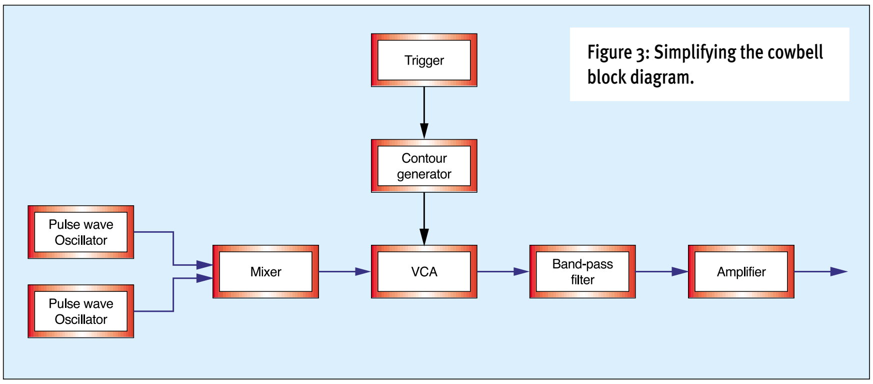


**Figure 3: Simplifying the cowbell block diagram.**

is Figure 3 (above).

I dug out a vintage drum machine to use as a sonic reference; the CR8000 CompuRhythm, another Roland unit of the same era and similar timbre that I prefer for its cowbells, claves and congas. By listening to the CR8000 cowbell and then sweeping two oscillators on my Nord MicroModular, I determined that the sound comprises a pair of tones with fundamental pitches of approximately 587Hz and 845Hz. With a frequency ratio of 1:1.44, these are suitably clangy, and serve Roland's purpose well.

At this point, it's worth complimenting Roland, because even small deviations from these pitches destroy the cowbell illusion. I would love to know how the company's engineers stumbled upon such an elegant solution.

Returning to the Nord, I selected the pulse waveform for both OscA1 and OscA2, and tuned them to the correct frequencies. Comparing the sound of the Nord to the CR8000, I noticed that the Nord was far too bright and 'synth-y' so, remembering that the higher harmonics of a triangle wave have a lower amplitude than those of pulse waves, I changed the waveform. This worked well, giving me an appropriately dull, clangorous tone.

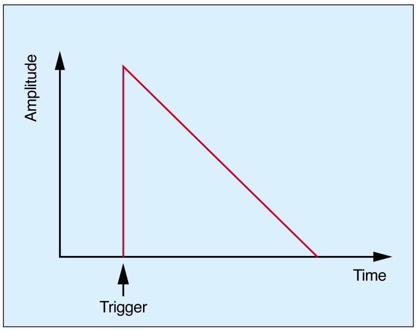


**Figure 4(a): A simple AD contour.**

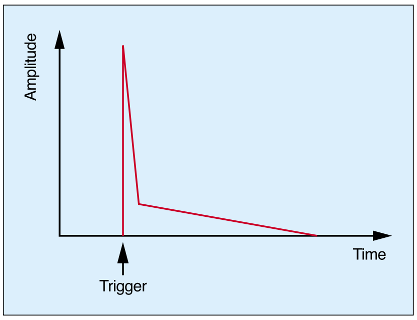


**Figure 4(b): The cowbell amplitude contour.**

A simple mixer then passed the combined signal to a multi-stage contour generator/VCA module. At first, I used a simple AD envelope (see Figure 4(a), below), but I could not achieve a satisfactory result with this. More listening suggested that the CR8000's contour comprised two stages: a high-amplitude, short-duration 'impact', followed by a more extended tail, as shown in Figure 4(b). I checked Roland's documentation, and this seems to bear out my observation. It says, "a series of R82 and C34 connected in parallel with C9 forms an envelope having abrupt level decay at the initial trailing edge to emphasise attack effect". See... told you so!

Having determined this, I chose a multi-stage envelope generator, Multi-Env1, inserted this after the mixer, and chose suitable values for the three-stage 'A/D1/D2' contour.

Moving on, I then added a band-pass filter, finding that a centre frequency of 2.64kHz worked well. The 12dB-per-octave option sounded a bit flabby, while the 24dB-per-octave cutoff shaped the sound too severely, limiting it to too narrow a band of frequencies, so I chose the 12dB-per-octave option and added a little resonance to accentuate the partials close to the centre frequency. This worked well, so I then passed the signal to an output amplifier, and added a clock generator to trigger the contour generator. Next, I sat back to enjoy my handiwork. In truth, the sound produced by Figure 5 is not identical to that generated by the CR8000, but it's close, and is every bit as valid as that produced by the Roland. What's more, a patch like this allows you to adjust the initial timbre, amplitude envelope and final tone colour, so in many ways it's far more useful than the predetermined sound coming out of the back of any preset analogue rhythm unit.

Now, here's a trick that I've used on many occasions to fine-tune imitative patches... I sample the sound I'm trying to emulate, and then replay it two or three octaves below its original pitch. This reveals many signal components that are of either too high a frequency, or too short a duration to be distinguished at the normal pitch. Performing this experiment on the CR8000's cowbell reveals no extra high-frequency information, but exposes a halo of noise surrounding the partials, particularly during the impact phase. I was able to recreate this on the Nord by adding a noise generator to the two oscillators, choosing a suitable 'colour', and mixing it in at a low level. I found that pink-ish noise (ie. one with the high frequencies suppressed) gave me the effect that I wanted, as shown in Figure 6 (on the next page).

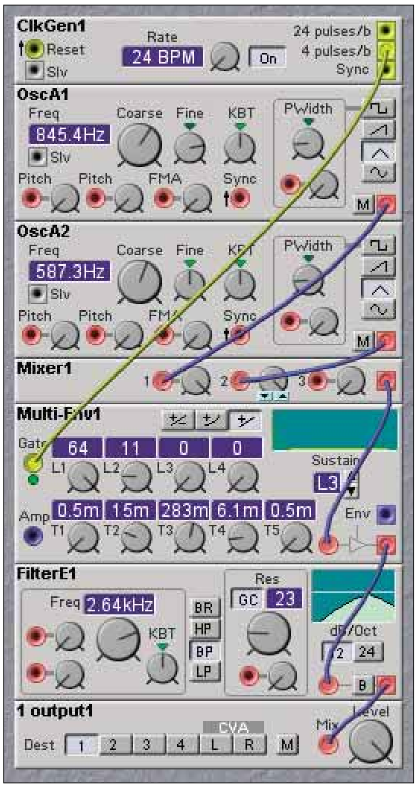


**Figure 5: Modelling a cowbell on the Nord MicroModular.**

## Recreating The Cowbell On An Analogue Synth

Do you remember the monster patch that I drew two months ago to show how many Analogue Systems modules were needed to recreate the TR808 cymbal circuit? (see *SOS* July 2002, or www.sound-on-sound.com/sos/jul02/articles/synthsecrets0702.asp) It was 23, which would also have required four rack widths of cases to house everything. This, in Analogue Systems-speak, is an

RS8000 synth, which costs about £2500 at UK prices... non-trivial expenditure in terms of finances, studio space, and your electricity bill. However, lest you think that *all* analogue percussion sounds require the same outlay, I am going to use an Analogue Systems Integrator to demonstrate how we can transfer the cowbell to a simple analogue synth. Well... when I say 'simple', there are certain constraints. Obviously, we need two, independently tuneable oscillators, so that precludes the ARP Axxe and Roland SH101 that I've been using for practical examples in recent months. We also require a band-pass filter, so, although the Minimoog satisfies the criterion for multiple oscillators, its single low-pass filter is inadequate for our purposes. That leaves the Korg MS20, but I've used that numerous times in recent months, and I fancy a change. To be fair to other manufacturers, I could have generated the following examples on a Doepfer, or one of the larger, more obscure modular synths, but I don't have one of those, and I *do* own an Integrator. So let's press on...

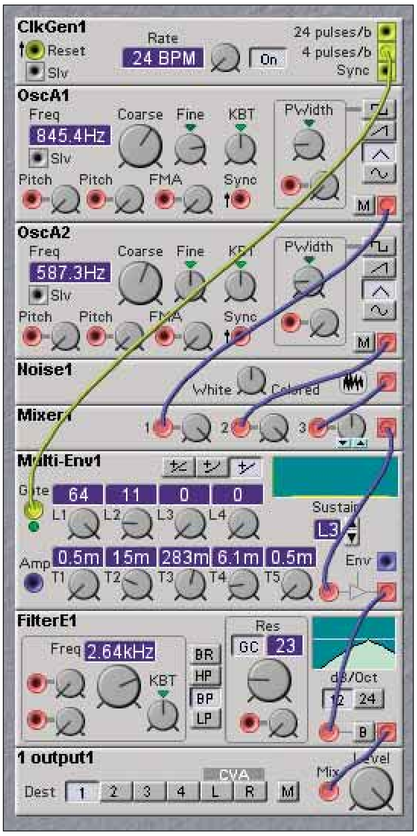

ClkGen1
Rate 24 BPM
24 pulses/b
4 pulses/b
Reset
Slv
On
Sync
OscA1
Freq 845.4Hz
Coarse Fine KBT PWidth
Slv
Pitch Pitch FMA Sync
M
OscA2
Freq 587.3Hz
Coarse Fine KBT PWidth
Slv
Pitch Pitch FMA Sync
M
Noise1
White Colored
Mixer1
1 2 3
Multi-Env1
Gate 64 11 0 0
L1 L2 L3 L4
Sustain L3
Amp 0.5m 15m 283m 6.1m 0.5m
T1 T2 T3 T4 T5
Env
FilterE1
Freq 2.64kHz
KBT
BR HP BP LP
Res
GC 23
dB/Oct 12 24
B
1 output1
CVA
Mix Level
Dest 1 2 3 4 L R M

Figure 6: Adding noise to the patch in Figure 5.

Firstly, we select two Analogue Systems RS90 or RS95 oscillators, and set them up as I did on the Nord; ie. with triangle-wave outputs at frequencies of 587Hz and 845Hz (see Figure 7, above).

Now we must mix the outputs from the oscillators. I found that I needed to add a little more of the higher frequency than the lower in order to get as close as possible to the CR8000 cowbell. This accentuated the upper partials of the sound, and gave it a bit more of the desired 'clank' (see Figure 8).

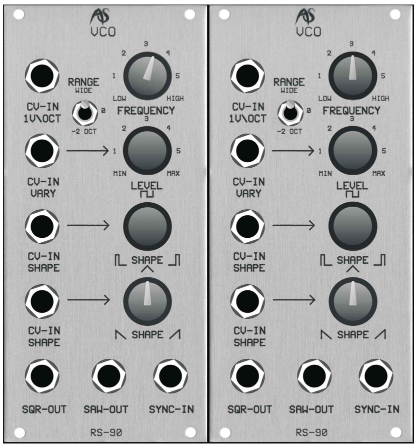


Figure 7: Setting up the oscillators.

Next, I set up the amplitude envelope and the VCA that it controls. Unlike the Nord, the Integrator has no multi-stage contour generator. This needn't be a problem, however — I can create the profile in Figure 4(b) using two contour generators and a CV mixer. However, this proved to be unnecessary. Experimenting with the VCA's linear and logarithmic CV inputs (annotated on the panel as 'CV1-In Lin' and 'CV2-In Log') and the Decay Time on the contour generator, I found that a short Delay applied at the linear input created an acceptable response. This is good... it saves two modules, two cables and, in practical terms, the need for a larger case for the extra modules (see Figure 9, top right).

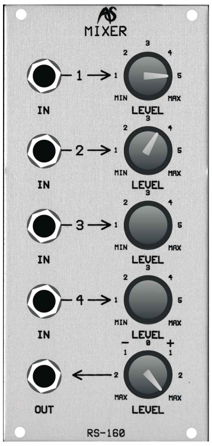


Figure 8: Mixing the oscillators.

Finally, I set the filter. As already noted, a low-pass filter is not suitable, because we need to remove low frequencies as well as high ones. So I chose the Integrator's RS110 Multimode Filter, inserted a cable into the Band Out output, and set the centre frequency and resonance to suitable values, as shown in Figure 10, right.

It's now time to see how it all hooks together. I directed the outputs from the oscillators to the mixer, setting the level for each as discussed. The output from the mixer passed to the VCA, which was controlled by the contour generator, and the shaped sound was then filtered before reaching the outside world. I have shown this in Figure 11 (on the next page), using blue 'cables' to show the audio signals, and black ones to show the single CV from the contour generator to the VCA. This is consistent with Figures 2 and 3. Once I had tweaked everything to perfection, I added a trigger to the contour generator's Gate-Trig In input, and my cowbell patch was ready to play.

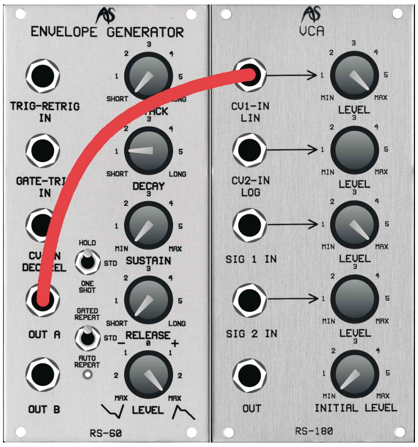


Figure 9: Setting the amplitude contour.

Before moving on, I must offer a word of warning... don't take the positions of the knobs in Figures 7 to 11 too seriously. I generated these diagrams in a graphics package and, while they are indicative of the correct settings, they are not precise. As always, small changes in settings can make a considerable difference to the result so, if have access to an RS Integrator, you should use your judgement to fine-tune the sound to your liking. In particular, it's vital that you should set the two oscillator frequencies as precisely as possible because, as I have already stated, even small deviations ruin the illusion.

Now, is there any way you can see to recreate this patch on a single-oscillator synth such as a Roland SH101 or ARP Axxe? If not, don't worry — I couldn't either, and assumed it to be impossible; but then I had a rare moment of inspiration, which I'd like to share. To be honest, I discovered the answer quite

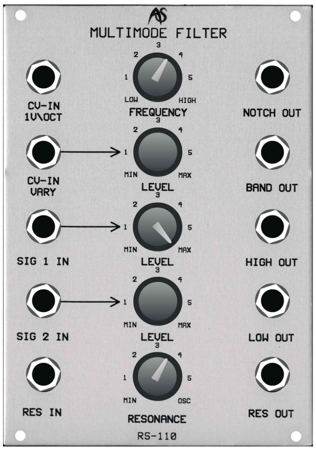


Figure 10: Setting up the filter.

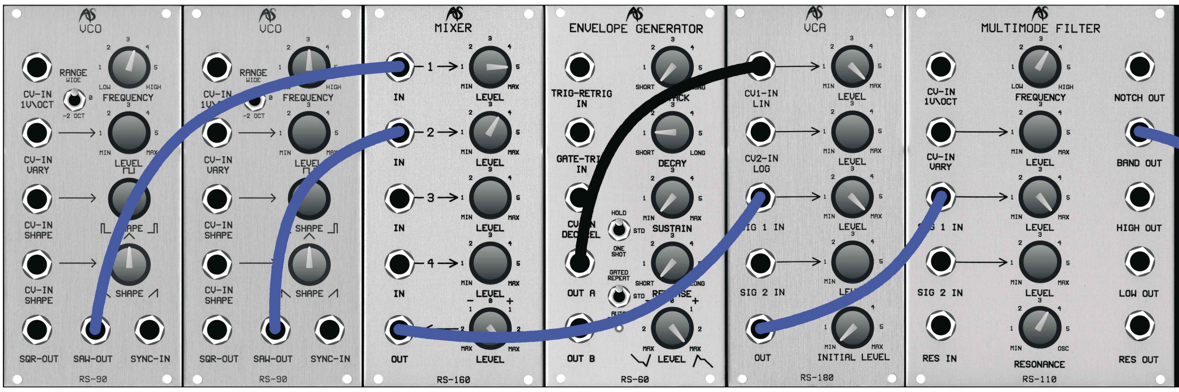


Figure 11: The RS Integrator cowbell patch.

by accident, but then serendipity is one of those things that make synthesis so rewarding.

The answer is; remove the 845Hz oscillator and make the filter self-oscillate at that frequency by turning its resonance to maximum. This, of course, creates a horrible howl, because in Figure 11, the VCA lies before the filter. Fortunately, the nature of this patch is such that there's no reason why we should not reverse the order of the filter and VCA. This means that we can have a single oscillator, a self-oscillating filter, a contour generator and a VCA producing all of the signal components and modifying them as before. Result... we need just four modules instead of six. What's more, an oscillating low-pass filter removes low- and high- frequency components from a signal, so it acts like a band-pass filter as well. Suddenly, the SH101, Axxe and Minimoog look quite capable of producing this sound!

The sound produced by this new patch is more percussive and more aggressive than before. If I tested you by playing this and the TR808 (or CR8000) next to other, and asking you which you preferred, I'm sure that you would pick the Integrator, and you would probably think that it was the Roland, because the other seems weedy by comparison. Damn... if only the Integrator had memories! This is a very nice patch.

Once I had discovered this, I was curious to see how far I could take my cowbell sound. I started by adding a second contour generator and a CV mixer to recreate the precise contour in Figure 4(b). I then added a little noise, and then a smidgen of reverb... Oh, what the heck. Do it yourself. Have some fun.

But what do you mean, you don't have an Integrator? Look at the simplicity of the architecture in Figure 12, below. You can create this sound on almost any synth that offers a triangle wave and a self-oscillating filter — *provided* that it allows a signal to pass through the filter while it oscillates. With carefully chosen filter settings, you might even get away with using a square or pulse wave as the initial waveform. Go on... try it.

## Claves

Now, before Synth Secrets leaves behind forever the electronic percussion of the late '70s, I want to discuss one more analogue rhythm sound with you. This is the 'Claves', which is my all-time favourite CR- and TR-series sound. Yes, I know that analogue kick drums are supposed to be the jewels in Roland's percussive crown, but I've never had that much time for them. By contrast, I still use the clave sound all the time. So let's return to the TR808 block diagram, and see how it all happens...

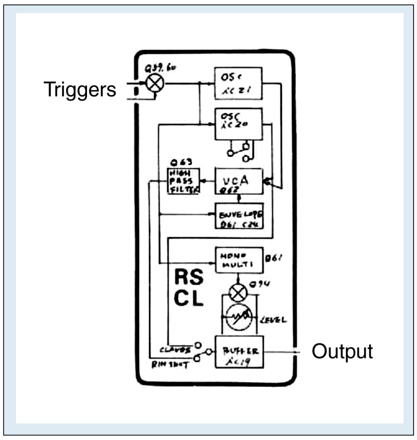


Figure 13: The TR808 Claves and Rimshot circuit.

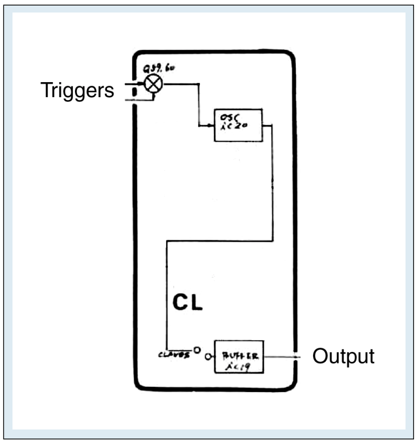


Figure 14: Removing the Rimshot circuit from Figure 13.

Figure 13 (above right) shows the architecture used by the TR808 for its claves and rimshot sounds. If you think that this looks confusing, I don't blame you; it is. So let's remove all the extraneous stuff (the parts relating to the rimshot sound), and see what's left (Figure 14, above). But now it looks *too* simple. How can a single oscillator and output amplifier produce the percussive sound I like so much?

The answer lies in our discussion of the TR808 kick drum back in February this year (see www.sound-on-sound.com/sos/Feb02/articles/synthsecrets0202.asp. To generate this sound, Roland used a type of oscillator called a 'Bridged-T' network, a circuit that, after a suitable trigger, produces a decaying waveform as shown in Figure 15 (above).

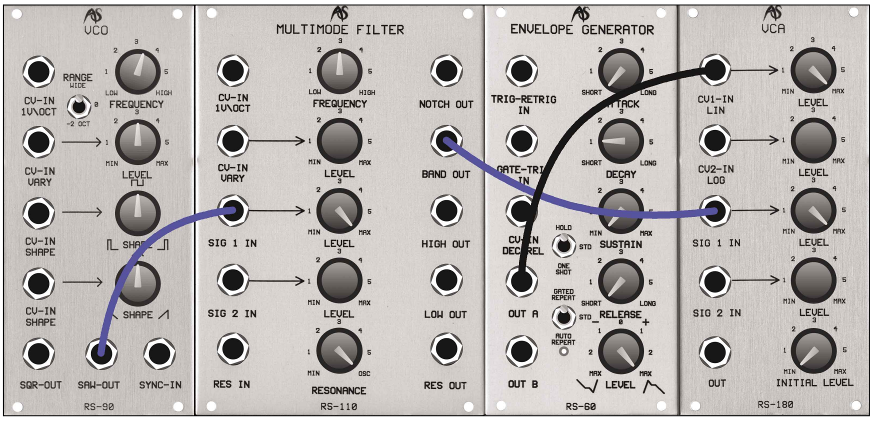


Figure 12: The simplified but improved cowbell patch.

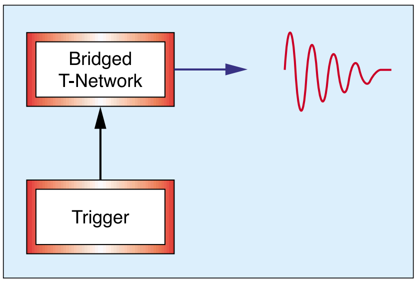


Figure 15: The response of a Bridged-T network oscillator.

This, then, explains the claves sound. It's a Bridged-T oscillator kicked by a trigger and allowed to decay without further interference. The signal is not even modified before being passed to the output. Simple!

Of course, conventional synths don't come with Bridged-T oscillators, so we must replace Figure 15 with a conventional combination of an oscillator, a VCA, and a contour generator. If we then add a filter to ensure that we hear the correct band of frequencies... well, I'll be hornswoggled, we've just recreated the patch in Figure 13, the block diagram for which appears as Figure 16 (shown right). All we now need to do is readjust the controls correctly, and we should obtain the claves sound.

Sure enough, sticking with the oscillator's triangle-wave output and choosing a suitable centre frequency for the filter (this time with zero resonance) we generate the appropriate initial timbre. We pass the output from this through the VCA, selecting an even shorter Decay time than before in the contour generator, thus limiting the sound to a nice, woody click.

To be honest, I'm very pleased with this patch. It's an almost precise recreation of the clave generated by my Roland CR8000, but it's more aggressive. And, once again, it's a sound that you can reproduce on simpler synths.

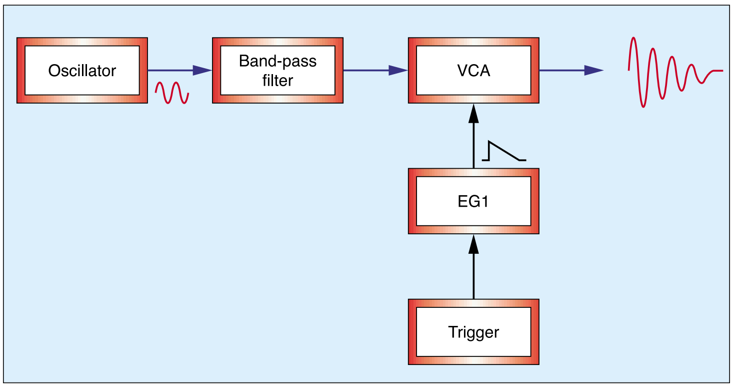


Figure 16: The cowbell and claves patch.

## Epilogue

And that's just about it as far as Synth Secrets and simple percussion goes. We've analysed and synthesized timpani, kick drums, snare drums, cymbals, hi-hats, bells, cowbells and claves, and that's enough to get anyone started. Sure, we could go on to recreate toms, rimshots, hand claps, congas, and any number of other percussion instruments, but many of these use the principles we have already discussed. So, next month, we're going to move on to an altogether different type of sound — pianos. SOS

# Synth Secrets

## Synthesizing Pianos

**Surely the only convincing synth pianos are sample-based ones? A sound as rich and expressive as that of an acoustic piano is far too complex to be rendered by subtractive synthesis... isn't it? We find out...**

Original photo courtesy of Yamaha

Gordon Reid

You might think that you'd have to be pretty wrinkly to remember the days when synthesizers were unable to produce convincing acoustic piano sounds. But do you? Ask yourself, what was the first electronic keyboard capable of sounding like, and responding like, a 'real' piano?

You could go back as far as 1954, and propose the earliest Wurlitzers, or to the mid-'60s, when the Fender Rhodes, Wurlitzer EP200, and Hohner electric pianos first appeared. But let's be honest... despite their electromechanical (as opposed to purely electronic) sound-generating mechanisms, none of these sounds like an acoustic piano. Each exudes *piano-ness*, and you can sometimes use them when a piano would be the preferred instrument, but try to play a Rachmaninoff piano concerto on one and you'll soon discover its limitations.

If you're sneaky, you might suggest the Mellotron, which featured a sampled piano among its more obscure tape frames. I used this on the Mellotron tribute album *Rime Of The Ancient Sampler*, and it fooled many listeners. Nevertheless, the Mellotron is unable to respond to velocity, and imposes its own character on any sound, so that's not an acceptable answer.

For obvious reasons, monophonic synthesizers are not admissible as piano substitutes. Neither, as you will know if you were ever forced to use one, were the host of ghastly electronic pianos of the early '70s. So let's now jump to the introduction of polyphonic synths in 1974. Hmm... still no good. The Yamaha GX1 couldn't sound like a piano, nor could the Oberheim 4- or 8-Voice. Moving forward another few years, we can also discount the Prophet 5, the OBX, the Jupiter 8, and all of the other big, analogue polysynths of the era. Well then, what of that hyper-expensive late '70s sampler, the Fairlight CMI? Sorry... its memory was too limited to hold and replay a convincing set of piano samples.

Let's move closer to the present, and enter the digital era. As we all know, the Yamaha DX7 was capable of remarkable imitations of *electric* pianos such as the aforementioned Fender Rhodes, Wurlitzers and Hohners, but even its phenomenal FM synthesis engine was incapable of generating *acoustic* piano timbres. Then, in 1984, the hyper-expensive sample players cracked it, when the Kurzweil 250 offered the best emulation yet heard — the famous 'Kurzweil Piano' on which the company's reputation is still based.

Two years later, this became affordable when the Roland HP5600, its stage counterpart the RD1000 (the 'Elton John' piano), and their rackmount sibling, the MKS20, became the first instruments to synthesize a satisfactory piano sound. Based on an early physical modelling concept (although nobody had thought to call it that at the time) they sounded remarkable. But, if the truth is told, their sound generators were still designed using samples.

Even as recently as 1987, no programmable synthesizer had produced a convincing piano sound. Then the Roland D50 introduced what we now know as S&S, or Sample and synthesis, which, for the past 15 years, has been the standard synthesis method across most of the industry. Next came the Korg M1, and some players liked its piano (though I hated it). But the real breakthrough was the Roland U20. Launched in 1989, its RS-PCM engine was an ancestor of Roland's JV sound engine, and its piano patch was almost as good as those produced by the most powerful samplers of the era. This was remarkable, because the U20 was a cheap, basic sample-replay keyboard with hardly a parameter to its name. But the samples were good, the looping was excellent, and the rudimentary envelopes did the job required of them.

By the dawn of the 1990s, S&S sound generators were *de rigeur*, and piano sounds began to pour forth from every £199 home keyboard. Nowadays, almost every synth offers a piano patch (or 20) and our ears have become so accustomed to them that many people have become unable to distinguish between a real piano and a synthesized imitation. Hang on... that's not true. I should have written that, nowadays, almost every *sample-based* synth offers a piano patch (or 20). Think of the exceptions... There are no acoustic pianos in the Clavia Nord Modular, the Korg Z1, the Waldorf Q, the Alesis Andromeda, the Novation Supernova II, or any other current synth that foregoes S&S. In fact, there has never been a convincing acoustic piano produced by subtractive synthesis, additive synthesis, or FM synthesis. Only samples appear to do the trick.

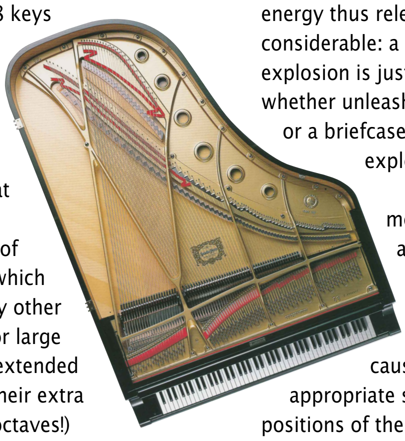
We've encountered this situation before. In August 2001's instalment of Synth Secrets (see www.sound-on-sound.com/sos/Aug01/articles/synthsecrets28.asp), I pointed out that it's not possible to create authentic-sounding acoustic guitar patches using subtractive synthesis. And therein lies a hint as to the nature of the problem. Sure, a piano hammers its strings rather than plucks them, but the two instruments exhibit some significant similarities. For one thing, piano strings interact with each other in different ways, depending upon their pitches and the number free to vibrate at any given time, just as happens on a guitar.

For another, each piano string interacts with a system that absorbs energy and then directs it back, exciting harmonics that may not be present in the initial waveform, just as happens on a guitar. Furthermore, the piano body and soundboard exhibit many resonances and anti-resonances that we cannot imitate using conventional equalisers or filters, just as on a guitar.

It's not a very encouraging scenario, I'll admit. Nonetheless, this month's task is to develop an understanding of the piano in order to synthesize it as best we can, so we'll start by taking a look at the piano mechanism itself...

## Piano Types

There are many types of piano, from Granny's unplayable Victorian upright, to the works of art that are the nine-foot grand pianos found in the more expensive concert venues. Most have 88 keys (which is why the largest synths and workstations use this number) although a few have more. This means that a typical piano has a fundamental range of over seven octaves, which is far greater than any other instrument (except for large pipe organs and the extended pianos, which, with their extra keys, reach to eight octaves!) Given that a young, healthy human can hear a range from approximately 20Hz to 20kHz — which is pretty much equivalent to 10 octaves — this means that a large piano covers as much as 80 percent of the range of human hearing!



An 88-note grand produces its sound using 88 hammers that strike nearly 250 strings. However, the natures of these strings differ depending upon where they lie in the range. At the bottom end, single strings are wrapped to high thickness, and the longest of these extend to seven feet or so. Next come notes produced by pairs of wrapped strings, then notes produced by triads (or 'tricords') of wrapped strings, and finally tricords of unwrapped strings, the shortest of which are just a couple of inches long. Given such radical differences in construction, it's not surprising that the tonality as well as the pitch of the piano changes dramatically from one end of the keyboard to the other.

As you can see in Figure 1 (above), the strings are suspended above a soundboard. However, despite being strengthened using struts, the soundboard is not a structural part of the instrument. This is because the pressure exerted by the strings can be as high as 20 tons, which would snap the board in an instant. Consequently, a heavy iron frame is used. Apocryphal stories exist of concert grands falling from a great enough height to cause the frame to collapse. The energy thus released is considerable: a 20-ton explosion is just as destructive whether unleashed by a piano or a briefcase of C4 plastic explosive.

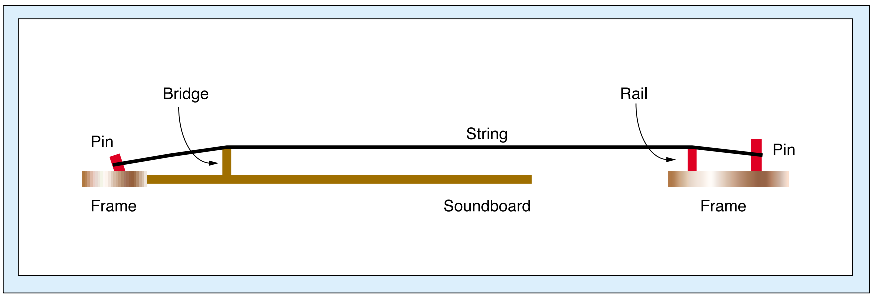


Figure 1: The suspension of a grand piano string.

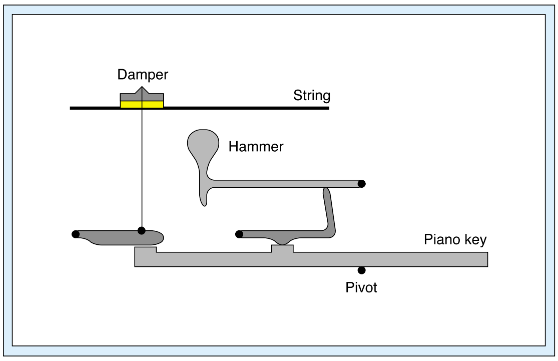


Figure 2: A simplified representation of a grand piano's mechanism.

Let's now look at the striking mechanism (see Figure 2 above). When you press a key, a system of levers (which I have shown here in greatly simplified form) causes a hammer to strike the appropriate string(s). If you look at the positions of the pivots and consider the movement of each part of the mechanism, you can see that a small amount of travel on the playing surface of the key translates into a very rapid movement of the hammer head. When this strikes the string(s), the energy in its movement is converted into vibrational energy in the string(s) themselves. Thus, it appears at first sight that the piano action is similar to the picking action used to play an acoustic guitar. However, it is very different, as I will now show.

## The Waveform

If you refer back to that August 2001 instalment of Synth Secrets, the one on how guitars produce their sound, you'll recall that, after you pick a guitar string, it vibrates at its fundamental frequency and overtones. I also mentioned that — because the plucking position cannot be a node of zero displacement — certain harmonics will be emphasised or eliminated depending upon the plucking position. Figure 3 (below) ▶

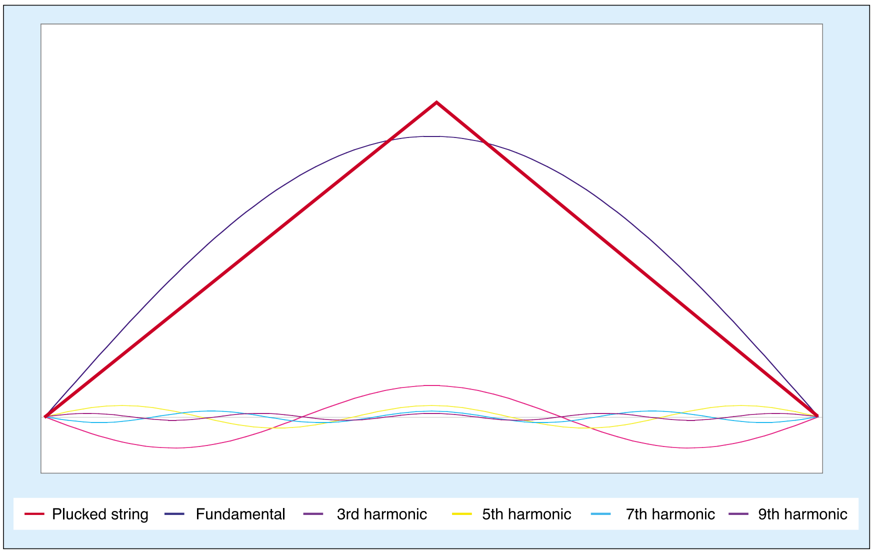


Figure 3: A string plucked at its centre has no even harmonics.

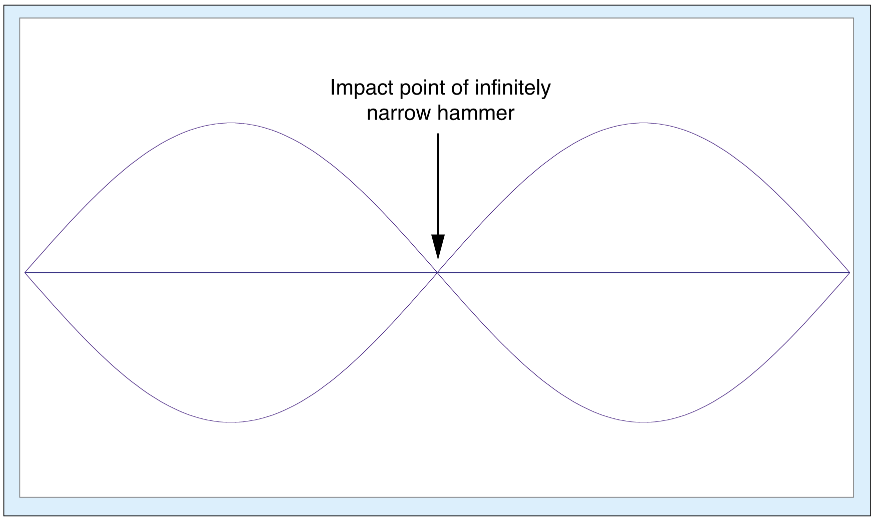


Figure 4: Hammering the centre of the string prevents the fundamental and other odd harmonics from being generated.

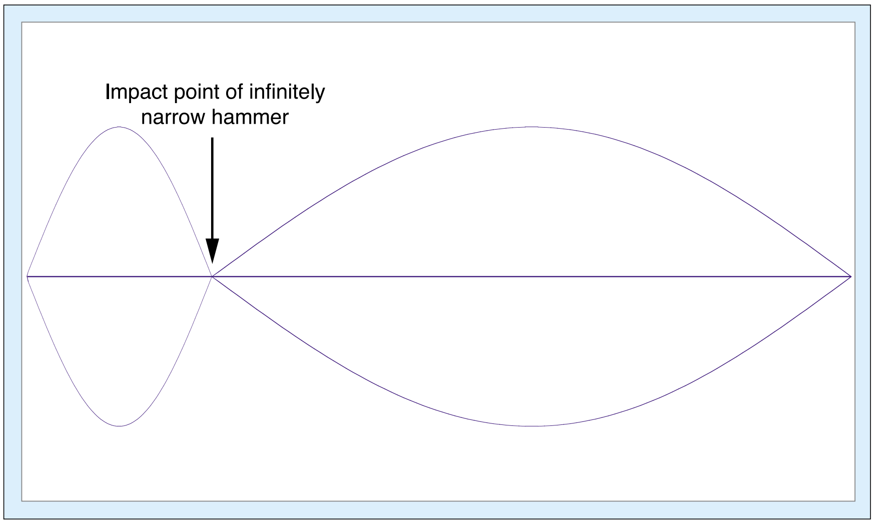


Figure 5: Hammering the string at an arbitrary point creates two notes of unrelated pitches (except by accident).

shows how a pluck in the centre of the string precludes the even-numbered harmonics from the initial spectrum.

Now, consider the piano string, which is not plucked, but hammered. Whereas the position at which a guitar string is plucked determines its *maximum* displacement, the piano hammer remains in contact with the string long enough to ensure that the position at which the string is struck is a node of *zero* displacement. If the hammer position were to be halfway along the string, you would, for an instant, have two sections whose lowest pitches lie one octave above the fundamental of the whole string. This means that the fundamental is missing from the resulting sound (see Figure 4, top). Likewise, hammering at the centre will ensure that the sound contains no third harmonic, or fifth, or seventh or ninth... or any of the other odd harmonics, all of which would need to exhibit displacement at this position.

The situation becomes even more complex if the hammer position is *not* halfway along the string. For example, if you hammer a piano string a third of the way along, you will divide it into two parts, one having a fundamental an octave above the other. But even this is simple compared to the situation where the hammer position is, say, 17.549 percent of the way along the string. As you can appreciate, the two pitches thus produced are unrelated to one another, and both are unrelated to the pitch of the whole string. I have shown the fundamentals of two such sections in Figure 5 (see above).

However, this situation lasts for just a fraction of a second. Soon after the hammer has completed its action, it bounces off the string. The vibrational energy is then free to run along the whole length of the string, and, by a convoluted mechanism of energy transfer, the string begins to vibrate at its fundamental and harmonic frequencies. However, the spectrum of the sound will still depend upon the striking point.

Given this, you would think that, to obtain a consistent tone, piano builders would position the hammers consistently from one end of the keyboard to the other. But this is not the case; you will find them anywhere from one seventh of the way along the string to about one 15th. This results in different initial frequency relationships, different interactions as the hammer leaves the string, and a different spectrum for the body of each note. What's more, these spectra change considerably when the strings are struck with different velocities, and the changes are not consistent from one end of the piano to the other.

Got a headache yet? Then get this... Most notes are generated by three strings, and it's all but impossible to tune these to the same pitch. You can get very close, to the point where any beating between them is almost undetectable, but the strings will soon become out of phase with one another, such that one string is moving 'up' while another is moving 'down', and so on. This leads to interference, with the strings swapping energy, reinforcing and at other times cancelling each others' modes.

Hmm... it's becoming fairly obvious that we can't fully analyse the vibration(s) in the string(s), but at least we can state how the energy from these vibrations is transmitted through the bridge to the soundboard. Except that we can't. Using different bridges can change the sound of a piano by a remarkable degree, and given that pianos generally have two of them — one for the treble strings, and one for the bass — and that they are coupled through the soundboard, we find that even this is too complex for us. What's more, piano soundboards have an irregular shape and are chamfered, so our previous discussions of vibrations in flat plates are, at best, approximations to the way that a soundboard vibrates. All I can tell you in the space I have here is that the modes of soundboards are enharmonic, and the way that they absorb energy from the strings and pass it back is far beyond the scope of Synth Secrets.

So where — in terms of subtractive synthesis — do we go from here? Sure, we can use three detuned oscillators to imitate the strings in a tricord, and add a couple more to create the atonal impact (see Figure 6 on the next page). We can even add contour generators and VCAs to crossfade between the two. But we will never be able to synthesize the complex interactions that give the piano its unique character. We could complicate Figure 6 by adding some sort of feedback to modify individual oscillator's waveforms and amplitudes, and this would no doubt generate interesting,

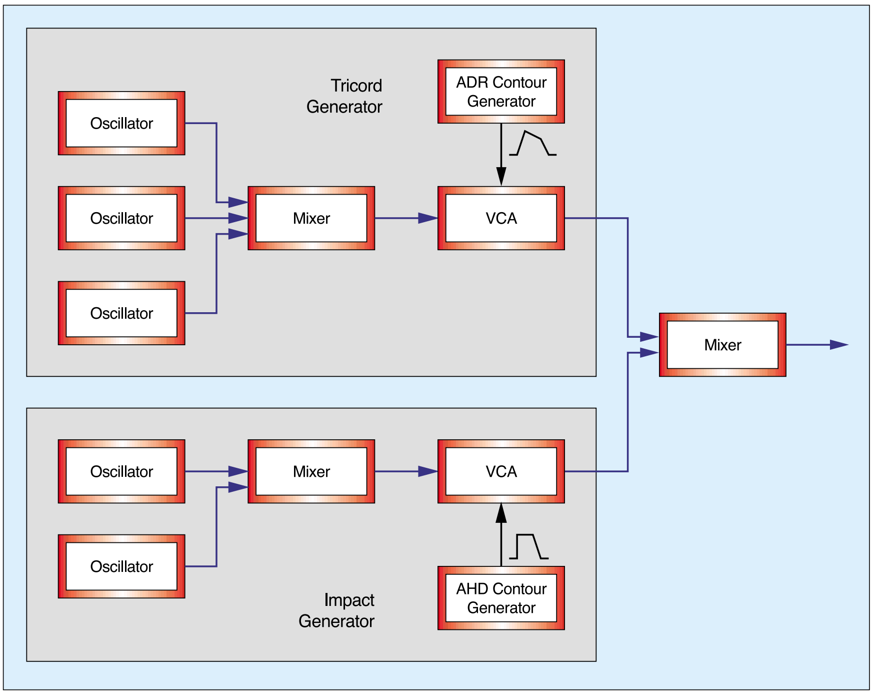


Figure 6: Imitating the tricord and the short-lived impact of the hammer against the string.

organic sounds. But it would not be the same as a piano, and it would not fool you. We don't even know what starting waveform to use for the oscillators!

## Amplitude Response

Many synths allow you to affect the loudness of a note using velocity sensitivity. And, although a piano's spectral response to changes in velocity can be very complex, its amplitude response is fairly straightforward.

Firstly, we know the general shape of the amplitude envelope for each note. There's an initial impact followed by a slow decay. We also know that if we don't release the note, the tail can linger for tens of seconds, which tells us that the rate of the decay diminishes as the note progresses. This is because, as the pairs and tricords interact, the rate at which energy is transferred to the soundboard (and is thence dissipated into the air) diminishes. I have shown the resulting envelope in Figure 7 below.

Of course, the piano has a large dynamic range, so we must make the amplitude contour velocity sensitive, as shown in Figure 8. What's more, the amplitude curves of low notes decay more slowly than those of high ones, so we should also make the contour sensitive to the note number or pitch CV, so that the decay shortens and maximum gain falls as the note rises (see Figure 9).

We can synthesize this using the architecture in Figure 10 (on the next page), which uses velocity sensitivity and keyboard tracking to affect both the maximum gain of a VCA and the decay rate of the contour that shapes it (in truth, there should be a handful of additional mixers and amplifiers, but I have omitted these for clarity).

## Brightness Response

Having created a simple model to synthesize the amplitude response, we must now consider the spectra of different piano notes, and how these change in time.

For any given note, we can separate the sound into three distinct stages: the initial hammer blow, the transition period during which the strings begin to oscillate harmonically, and the tail, which is dominated by the fundamental, and whose higher harmonics decay most rapidly.

I have accounted for the first of these stages in Figure 6 by adding the two oscillators in the Impact Generator. If both are tuned higher than the note itself, and the AHD contour generator keeps their contribution brief, they will make a suitable

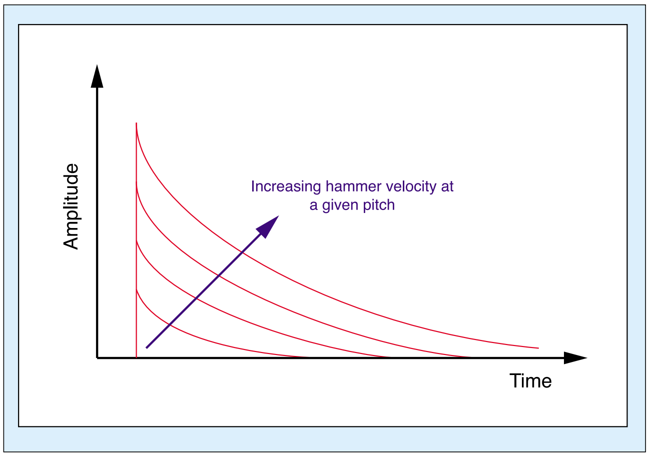


Figure 8: The decay curve at different hammer velocities.

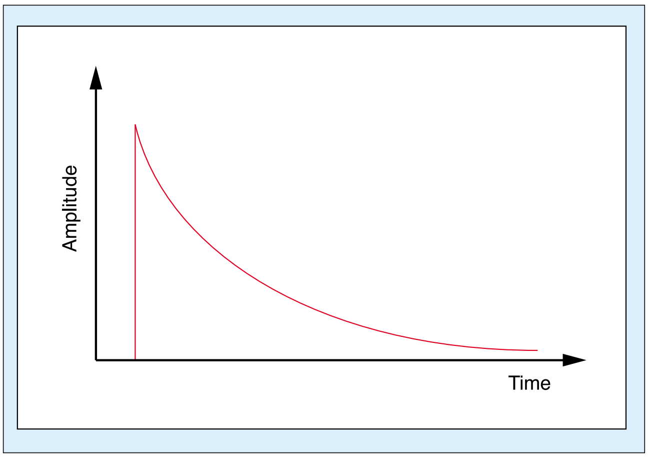


Figure 7: The decay curve of a typical piano note.

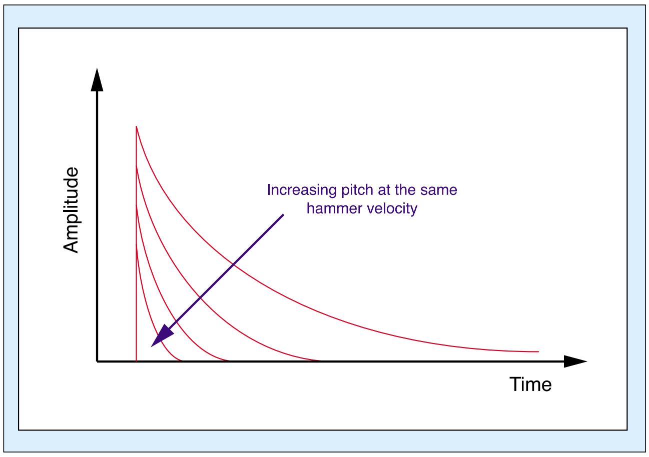


Figure 9: The decay curve at different pitches.

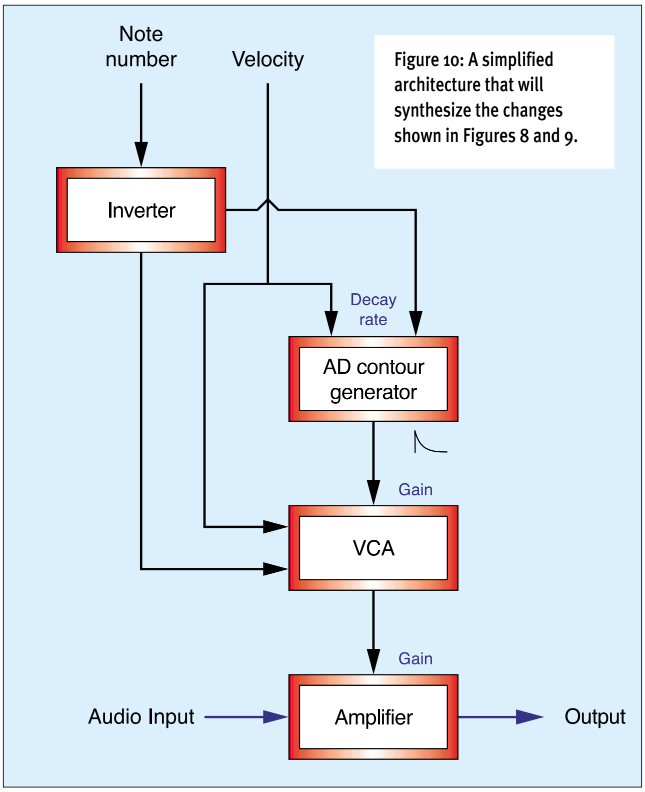


Figure 10: A simplified architecture that will synthesize the changes shown in Figures 8 and 9.

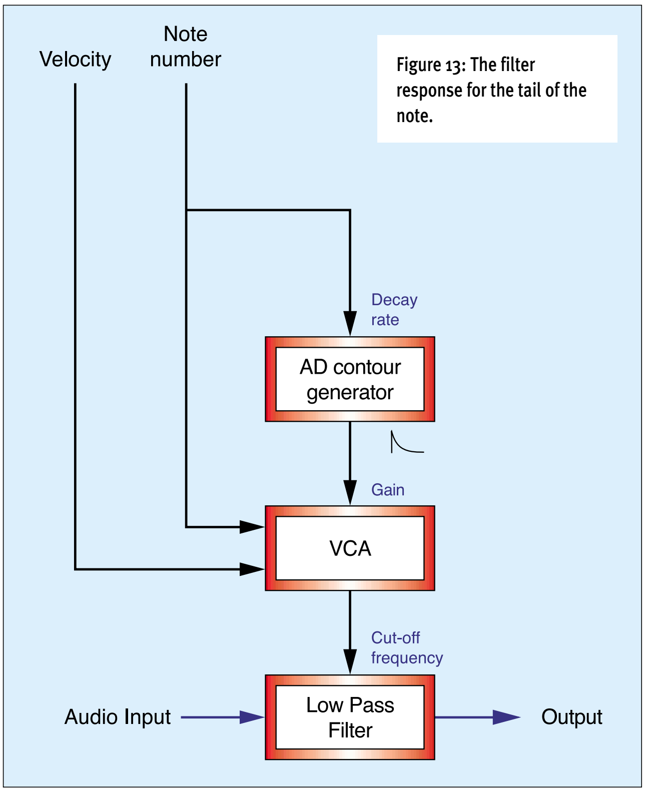


Figure 13: The filter response for the tail of the note.

‘chink’ at the start of the note. For completeness, we could also add a bit of tuned noise to mimic the mechanical clunk that accompanies the hammer blow itself, but I’ll leave this to you.

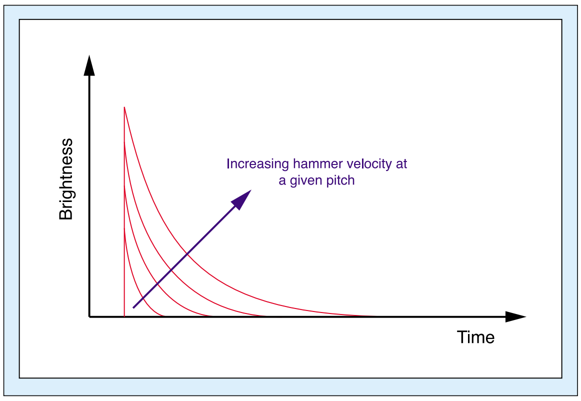


Figure 11: Notes hit harder are brighter.

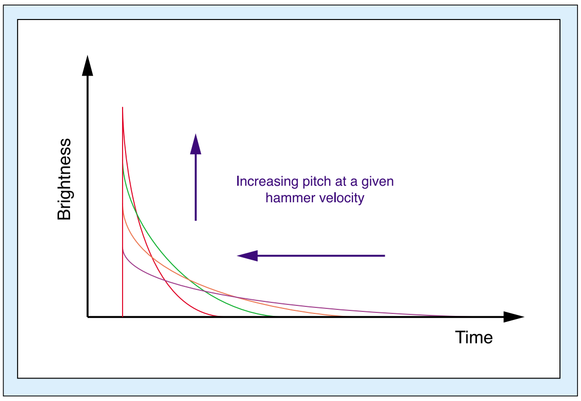


Figure 12: Higher-pitched notes start brighter, but dissipate their energy more quickly.

The second stage is much harder to emulate, because it is here that the nature of the waveforms is changing most rapidly. I suppose it’s possible that we could invent a synth architecture to imitate this, but I know of nobody who has succeeded, and (even if possible) the block diagram would take up more pages than are left in this month’s copy of *Sound On Sound*. We’ll simply have to return to this stage later. So let’s move to the third stage, which is the simplest to reproduce, requiring (to a first approximation) little more than a contour generator and a low-pass filter.

Experience tells us that piano notes are brighter when hit harder (see Figure 11) so we will need to scale the filter appropriately. Furthermore, piano notes are brighter at high pitches than they are at the low ones. But remember that the high-frequency energy (indeed, all the energy) of higher-pitched notes is dissipated more rapidly, so we observe the curious result that is Figure 12 (left).

This means that the filter cutoff frequency must respond to note number (or pitch CV), key velocity, and some form of contour, with the decay rate of the contour responding to the note number (see Figure 13, above).

If I wanted to complicate matters, I could point out that, for the lowest notes on a grand piano, the fundamental pitch has very low amplitude, and the note that you *think* you hear is to some extent implied by the harmonics. This suggests that we require a high-pass filter for the lowest notes in our synthesized sound. But I think we should ignore this. There’s only so far we can go before we reach the point of diminishing returns, adding complexity for little extra benefit.

## Tuning

Next, we must turn to the subject of tuning. I mentioned in that instalment on the physics of the acoustic guitar that a string’s harmonics are *stretched* as the pitch increases and/or the excitation increases in amplitude. This is because the string requires a finite length in which to bend over the bridge and nut, thus shortening the effective length. Not particularly important at the fundamental frequency, this becomes more significant at higher harmonic numbers, and stretches the harmonic series from 1:2:3:4:5:6... to something that may look more like 1:2:3:4.01:5.02:6.04... and so on (don’t take these numbers too seriously; I just made them up to illustrate the point).

This *stretching* of the harmonic series has far greater consequences on a piano than a guitar because, on the piano, we are able to play chords reaching across seven octaves. If you play a concert-pitch 'A' on a piano (440Hz) simultaneously with the 'A' two octaves above, the frequency of the fourth harmonic of A440 should be equal to the fundamental of the upper 'A'. However, because A440 exhibits stretched harmonics, the upper 'A' will, if tuned to 1760Hz, sound a fraction flat! Indeed, the human ear/brain is so accustomed to this that a perfectly tuned piano not only sounds out of tune, it sounds dull. A stretched tuning system makes the overall sound 'sweeter' to our ears, and we need to imitate this. We do this by making the oscillators track the keyboard at a ratio just a fraction greater than 1:1. We can do this by placing a near-unity amplifier in the pitch CV path (see Figure 14, right).

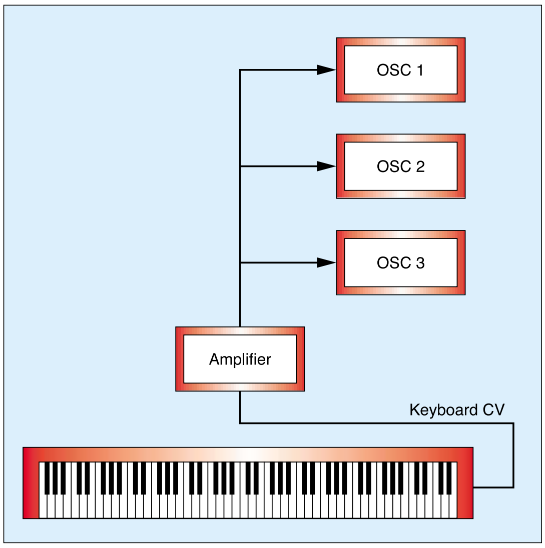


Figure 14: Stretching the keyboard CVs.

## Putting It All Together

If I now combine all the elements discussed above into a single patch (see Figure 15, below), you can see that it's a biggie which exceeds the capabilities of most synths — even without a complex oscillator section to recreate the authentic piano tone! And, even when carefully programmed on a large modular synth, it sounds like a Wurlitzer or Pianet. It does *not* sound like a Bösendorfer.

This is hardly surprising, and it would still be the case even if we modelled the hammers and strings perfectly. This is because we have totally overlooked the resonances of the soundboard and the glorious reverberant effects of the piano body itself.

What's more, we haven't even touched upon complicating factors such as the action of the pedals. The most important of these is the right-hand 'sustain' pedal, which lifts the dampers so that all the strings can vibrate simultaneously. If you strike a single note with this pedal depressed, the hammer excites the strings that produce that note, but the energy then passes through the bridge and soundboard to excite other strings. Some will vibrate *sympathetically*, because they share modes of vibration with the initial note, while others will not. However, after a few fractions of a second, the interactions of the strings, the soundboard, and the bridge become so complex that it is impossible to calculate which strings will be vibrating and with what amplitudes, spectra and phases. The sound thus produced is extremely complex and is, for obvious reasons, called *sympathetic resonance.*

So... is it impossible to create an acoustic piano patch on an analogue synth? The strict answer is 'yes', but as someone who performed for a couple of years using a 76-note analogue polysynth as a stage piano, I have to admit that the situation is not as gloomy as it seems. Sure, the patch I used would never fool you into thinking that you were listening to a real piano, but, like the Fender Rhodes, Wurlitzers and Hohners mentioned at the start of this article, it was piano-like, and useable for rock and roll. The polysynth in question was the Roland Super JX10, and next month, I'll show you how it managed it.

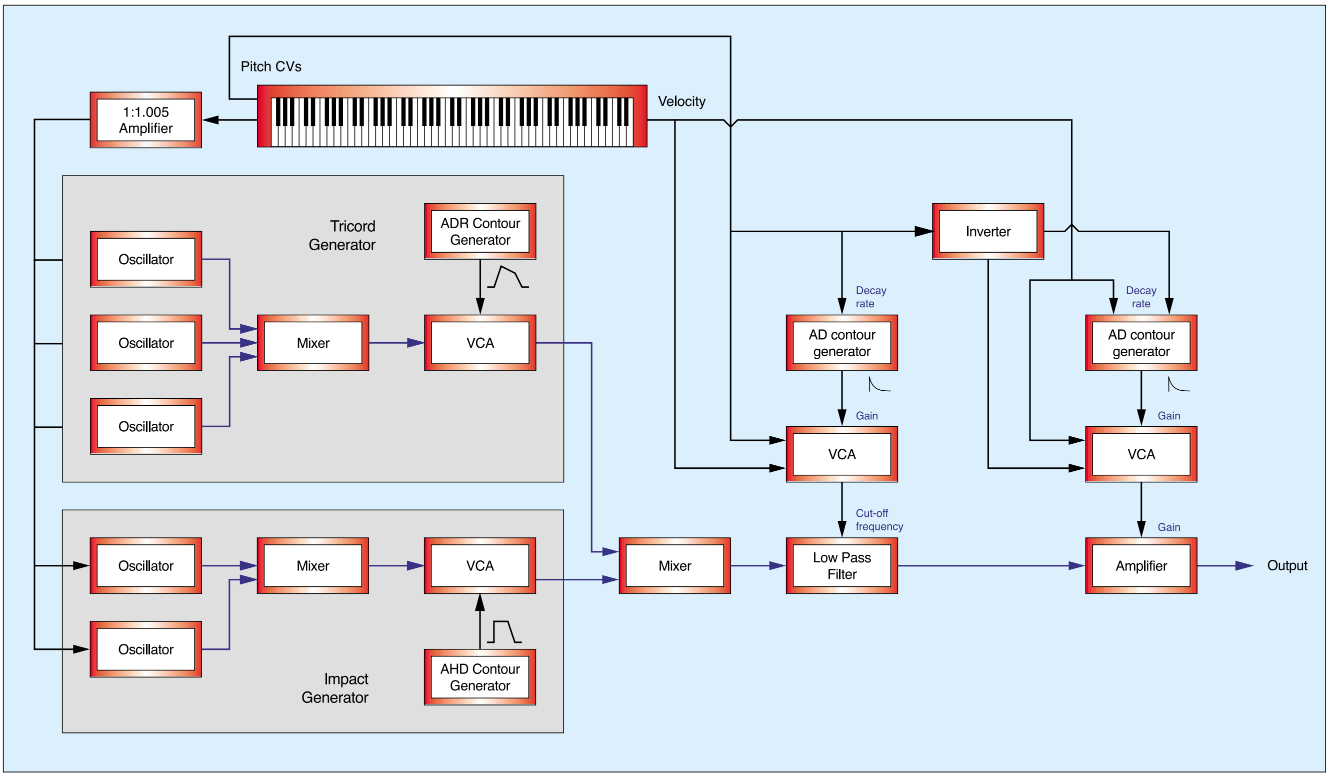


Figure 15: Trying to create a piano sound using a realistic number of modules.

# Synth Secrets

## Synthesizing Acoustic Pianos On The Roland JX10

**As explained last month, synthesizing the sound of an acoustic piano is difficult, but it can be done reasonably realistically, as the 1986-vintage Roland JX10 shows. We find out how Roland managed it...**

Gordon Reid

Last month, I concluded my discussion of the acoustic piano by promising to show you how the not-very-humble (and highly underrated) Roland Super JX10 analogue polysynth can create a satisfactory piano sound. And I will. But if you're expecting a simple set of instructions suggesting that you patch the output of the voltage-controlled doodah to the input of the exponential wotsit, you're barking up the wrong sequoia. That's because — before we're in a position to do so — we must first investigate an area of synthesis not yet covered in Synth Secrets.

Oscillator synchronisation (or 'sync') has been around since the birth of analogue synthesis. Nevertheless, it's one of the least understood facilities on any synthesizer. To be honest, that's not surprising... it's a non-linear operation, and it comes in at least three flavours. But given our well-defined target (ie. a decent analogue emulation of the piano sound) I'm going to narrow our focus to concentrate on the variety and implementation that will best help us to achieve this. It's the most common form, and it's called hard sync.

### What Is Hard Sync?

Unless this is the first copy of *Sound On Sound* you've ever held, I'm sure that you'll have read at least one review or retrospective of an analogue or virtual-analogue synth containing the cliché that it's "capable of tearing sync sounds". Some of us (I fear that I should hang my head in shame) have even been guilty of writing it. Indeed, it's quite possible that you've spent far too many hours twisting the pitch knob of a sync'ed oscillator to make it go 'zeeeooooww'. But what, precisely, is happening to the waveform as you do so?

Figure 1 shows the output from a perfect ramp wave oscillator. It's not an analogue, software-generated, digital, or any other type of ramp-wave oscillator... it's just a representation of a perfect ramp-wave oscillator, and it has a frequency of some arbitrary value, which I'll call 'F'. We will also call the waveform the *slave* waveform from now on, because this is the signal that will be affected by the operations we perform.

Now we're going to consider a second waveform — a perfect square wave — with a frequency of twice F. I've shown this in Figure 2. This is the *master* waveform, because it is the one that affects the slave.

Next, let's assume that, by the magic of electronics, we can extract a series of triggers from the master waveform, and that a trigger occurs each time the master wave completes a cycle. The resulting triggers occur at each of the positions shown in Figure 3. We're now going to use these triggers to perform an innovative trick: we'll reinitialise the slave waveform (or, to be a bit more scientific, reset the phase of the slave to 0 degrees) each time a trigger is encountered, as shown in Figure 4 (above right).

In this case, the result is another ramp

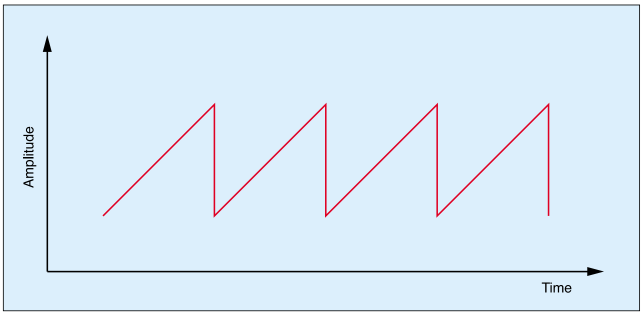


Figure 1: A perfect ramp wave.

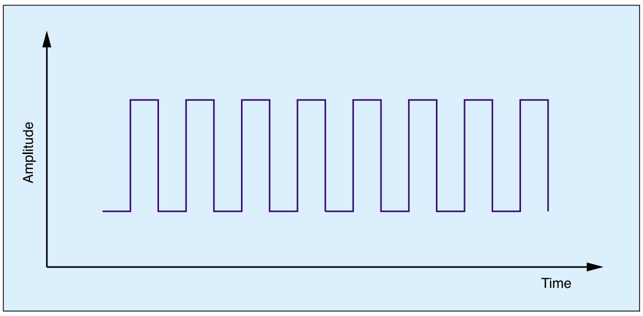


Figure 2: A perfect square wave.

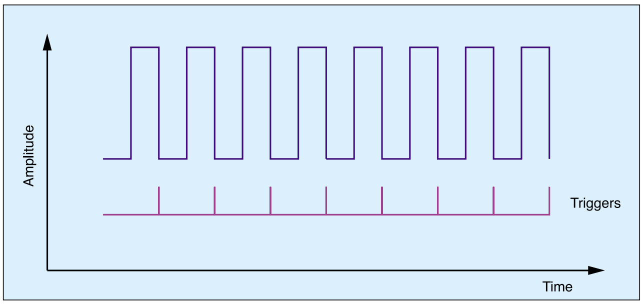


Figure 3: Deriving triggers from the master waveform.

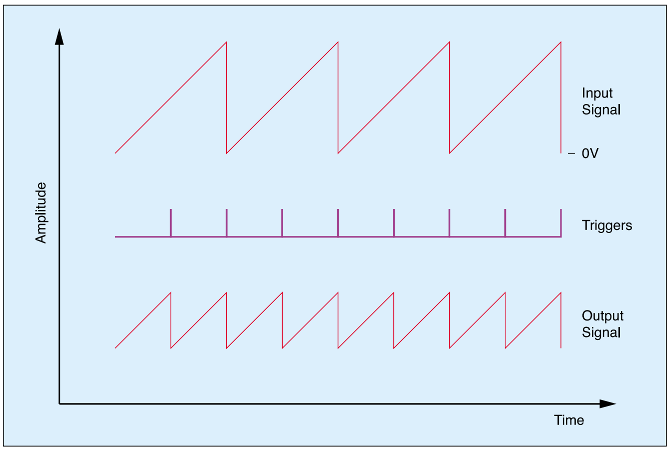


Figure 4: Reinitialising the slave at a frequency of 2F.

wave. Although it's generated by the slave oscillator, it has the same frequency as the master, but half the amplitude of the original slave. If this seems to be a rather arcane way to double the frequency and halve the amplitude of a signal, I agree. It's not very interesting, but let's persevere…

At first sight, it seems that little happens when we alter the pitch relationship between the master and the slave. To illustrate this, I will change the frequency of the master from 2F to (8/3)F, as shown in Figure 5 (below). The result is much the same as before; we have obtained another ramp wave with the same frequency as the master, but this time with even lower amplitude. Boring!

If you consider Figures 4 and 5 again, you might see the reason why we are obtaining such uninteresting results. It's because, up to this point, we have considered two cases where the master frequency is higher than that of the slave. So let's now consider a couple of instances where the slave is set to the higher frequency.

Figure 6 (on the next page) shows such a situation, with the master running at 0.8F. In this example, the slave has time to complete a whole cycle, plus a little bit, before it is reset. The resulting waveform is much more interesting than before, with an unusual harmonic structure that you cannot easily obtain by other methods. Nonetheless, the output frequency is again the same as the master. Indeed, no matter what the relative frequencies of the two oscillators may be, the output frequency will always be equal to the master frequency. This is the first rule of hard sync:

*When two oscillators are hard-synchronised, the pitch of the output is equal to the pitch of the master.*

But this isn't the end of the story. Let's consider what happens when we increase the slave frequency further with respect to the master.

As you can see in Figure 7, a ramp wave of higher pitch is able to complete more of its cycle, or more cycles, before it is reset, thus creating a different waveform. As always, the pitch is that of the master, but the tone is different from that shown in Figure 6. So here's the *second* rule of hard sync:

*When two oscillators are hard-synchronised, then if the master frequency is lower than the slave frequency, changing the pitch of the slave changes the timbre of the output.*

▶

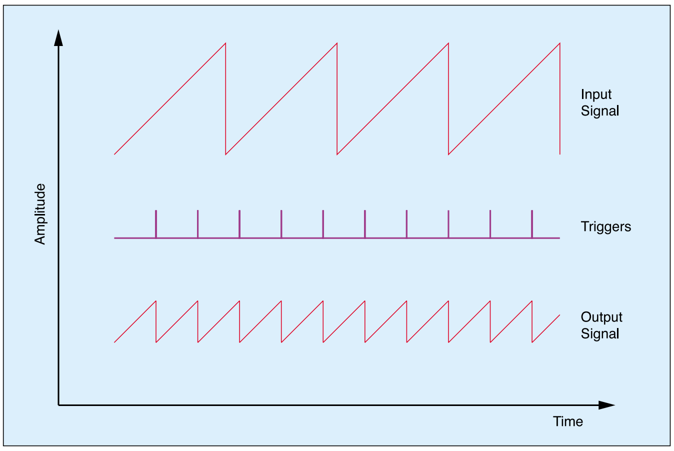


Figure 5: Reinitialising the ramp wave at (8/3)F.

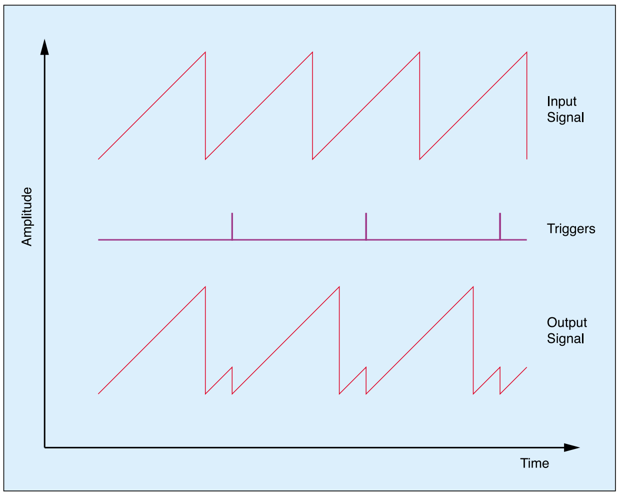


Figure 6: Synchronising an oscillator of frequency F using a master of 0.8F.

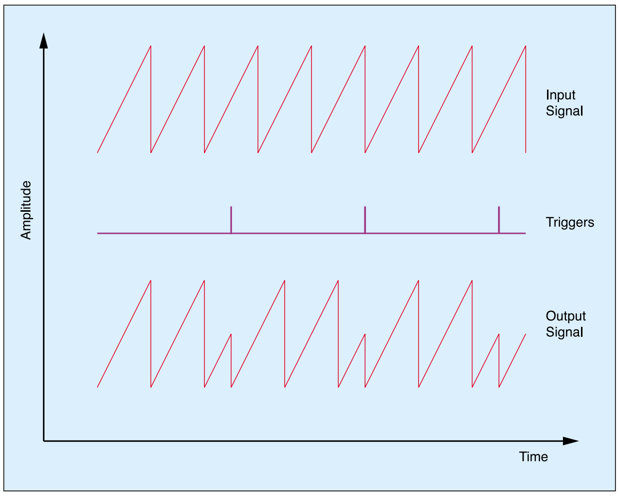


Figure 7: Sync'ing a ramp wave slave of higher frequency than before.

These changes are not subtle. The output is a ramp wave whenever the slave frequency is a whole number multiple of the master frequency, but at any point in between, strange-sounding variations on the slave waveform are created, and the harmonic structure of the slave is similarly affected. So if you sweep the slave frequency up or down, the output of the sync'ed oscillators changes constantly, from ramp waves through many exotic variants, back to ramp waves again, and continues to change for as long as you keep changing the frequency of the slave. This is what gives hard sync its ability to generate such distinctive sounds.

Before moving on, I should mention that, although these examples have used ramp waves as the slave, there's no reason these days why we can't use other waves. But in practice, there used to be an important reason. The electronics in analogue synths generally limited the use of hard sync to ramp and pulse waveforms. With modern digital synths, which offer greater flexibility with regard to their waveforms, you can obtain a wider range of effects, including tonal changes when the master frequency is higher than the slave frequency (see the 'More On Oscillator Sync' box below)

## Generating Hard Sync

We can recreate all of the waveforms and sounds we've discussed so far using just two oscillators, provided that the intended slave has a 'sync' input. The Analogue Systems RS95 VCO shown in Figure 8 here is a good example of such an oscillator, and it will generate hard sync provided that you apply a square (-ish) signal of suitable amplitude to the 'Sync In' socket in the lower right-hand corner of the panel.

Figure 9 (on the next page) shows how we connect two of these oscillators to generate and play sync'ed sounds. Firstly, we must patch the keyboard pitch CV to the pitch CV input on the master oscillator, whereupon the first rule of hard sync, as stated earlier, allows us to play the patch in standard fashion. We then direct the square-wave output from the master oscillator to the 'Sync In' of the slave. The circuitry within the slave converts the master waveform to the series of triggers described in Figure 3, and these reinitialise (or 'synchronise') the slave waveform as I illustrated in the subsequent examples. Taking the signal from the slave's sawtooth output allows us to hear the complex waveforms generated. Figure 10 shows this patch in standard Synth Secrets format.

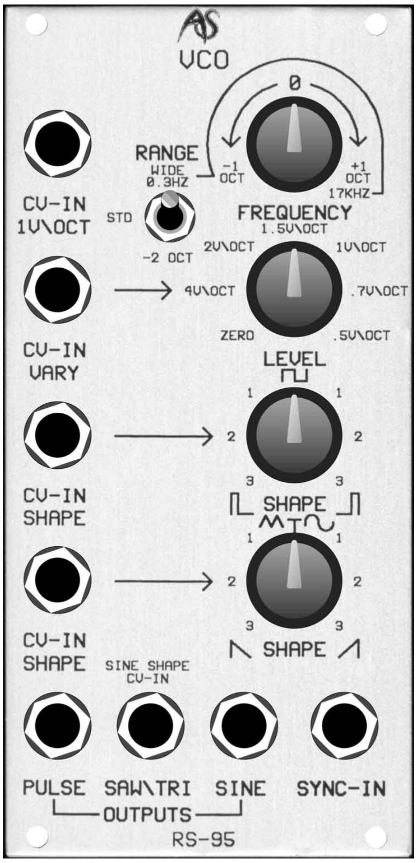
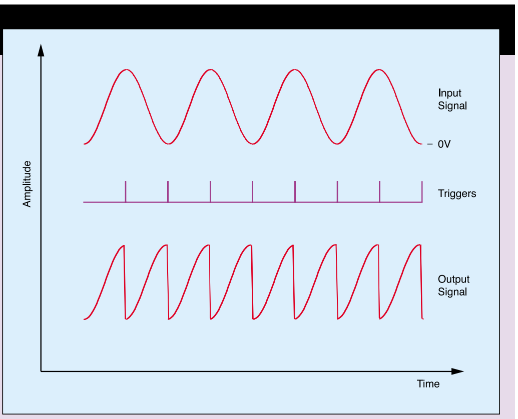


Figure 8: An oscillator with a Sync input.

The sound produced by Figure 10 is interesting for two reasons. Firstly, it has a timbre quite unlike the square and sawtooth waves used to generate it. Secondly, only the master is tracking the keyboard, so the frequency relationship between the master and slave is different for each note, resulting in different tones for each. Nonetheless, the timbre remains static within a single note, and if this were the limit of sync's capabilities, we would require additional tools such as dynamic filters to inject some life and tonal

## More On Oscillator Sync

**The first few examples in this article suggest that you cannot generate new timbres when the master frequency is higher than the slave frequency. This is not strictly true, although it is true when the slave waveform is a ramp or square wave. Fortunately, some synths allow us to synchronise different waveforms, and it is easy to show that if we use an alternative (such as the sine wave in the diagram on the right) we can create new waveforms even when the master waveform is at the higher pitch.**



Sync'ing a sine wave of frequency F using a master of frequency 2F.

variation into the sound.

However, it's possible to overcome this by making use of the *second* rule of hard sync mentioned earlier, and setting up (for example) a triggered contour generator to sweep the pitch of the slave (see Figures 11 and 12 below). If you set the ADSR correctly (please ignore the positions of the knobs in Figures 9 and 11... they are all set to 12 o'clock by default) the pitch of the slave is modulated to create dramatic sweeps of tone each time you press a key. In fact, if you offset the pitches of the two oscillators correctly, and apply the right amount of gain to the sweep signal, you will recreate the powerful sync sounds obtained from vintage synths such as the Moog Prodigy.

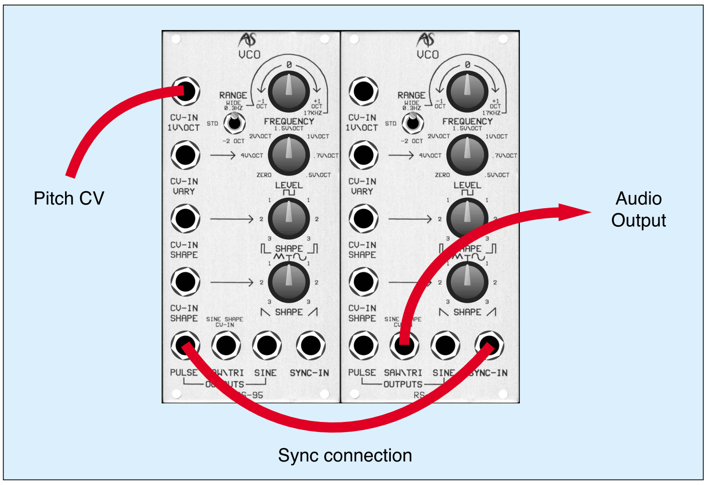


Figure 9: Physically sync'ing one RS95 oscillator to another.

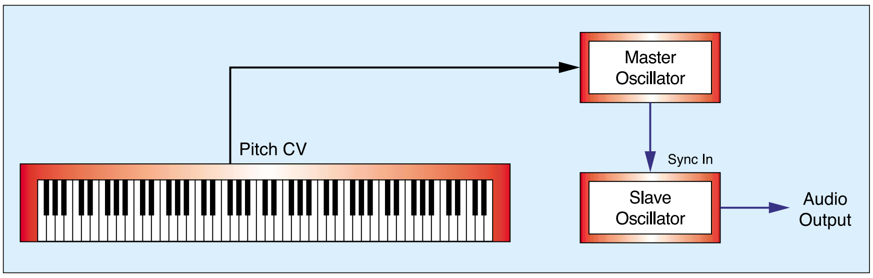


Figure 10: A simple 'sync' patch.

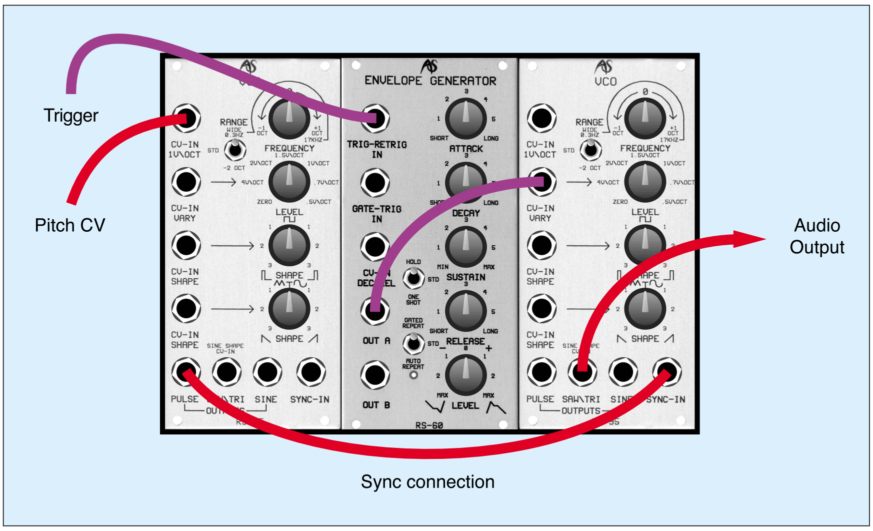


Figure 11: Patching a swept sync sound.

Of course, there's no reason for you to limit the sync modulation to a single ADSR contour generator, and with a more powerful pre-patched (or modular) synth, you will be able to use all manner of modulation sources. You need only look at the front panel of an ARP Odyssey to see what is possible; pitch modulation of the slave can be provided by an LFO, an ADSR envelope, S&H, the mixed VCO1/VCO2 signal produced by the S&H mixer, noise, and even an external CV... plus numerous combinations of these. The possibilities are enormous, and this is just one reason why players are still able to coax new variations of sounds from the Odyssey — a synth that celebrated its 30th birthday this year!

It might now seem that we're a long way from our original quest to synthesize the acoustic piano. But if you cast your mind back a month, you'll remember that there were two primary reasons why we were unable to create a convincing piano patch using conventional oscillators, filters, and so forth. One of these was because we could not imitate the natural resonances and reverberations of the piano soundboard and body. The other was that, although we used contour generators, filters and amplifiers to mould the overall shape of the sound, we were unable to recreate the dramatic timbral changes that occur at the start of every piano note.

Happily, we now have a powerful tool to help us overcome this, because hard sync is capable of much more than the aforementioned 'zeeeooooww' sounds. It is one of the easiest ways to imitate the sound of a hammered or plucked string, and it's used in some of the most evocative harpsichord and clavinet sounds ever produced by an analogue synth. But I'm getting ahead of myself...

## Synthesizing The Piano Timbre

I have already stated that the Roland Super JX10 is capable of producing a useable piano sound. I know this from experience, because I used Roland's factory 'H1: Acoustic Piano' Performance as a stage piano for a couple of years. Falling somewhere between my RMI Electrapiano and the pair of

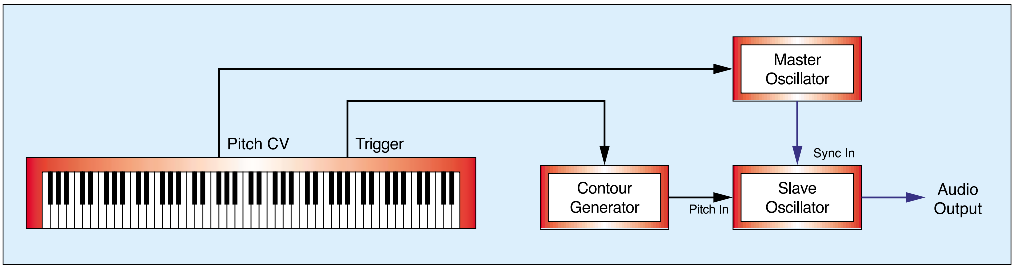


Figure 12: Creating a swept sync sound.

▶ MKS20s that I eventually adopted, it had a character of its own, and contributed to tracks that might have ended up quite different had they been written with a 'real' piano to hand.

As you may be aware, the JX10 is essentially two Roland JX8Ps in a single box, with a bunch of extra parameters that allow you to combine two JX8P 'Tones' (which we would normally call 'patches') into a single 'Patch' (which is what most people would call a 'performance'). It is a hybrid synth, with DCOs (Digitally Controlled Oscillators) and the quantisation of parameter values that is necessary if patches are to be stored in memory (to understand why quantisation and memories go hand in hand, please refer back to Synth Secrets 21, in *SOS* January 2001, or at www.sound-on-sound.com/sos/jan01/articles/synthsec.asp). The JX10 utilises a 'digital parameter access' programming system in which every voice parameter has a number and an associated value.

The tables elsewhere on this page show the oscillator settings for the 'Piano 1-B' factory Tone, which comprises half of the 'Acoustic Piano' Patch. Unless you're practised at reading tables of this nature, they may not mean much at first sight, so let's sort them out…

The first table (below) shows the settings for Oscillator 1, also known as DCO1. This is a simple oscillator with controls for octave range, waveform, tuning (in semitones), LFO modulation depth and envelope depth. As you can see from the table, its output is an 8' square wave with no tuning offset or modulation.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| **DCO1** | | |
| **11** | **Range** | **8'** |
| **12** | **Waveform** | **Square** |
| **13** | **Tune** | **0** |
| **14** | **LFO Depth** | **0** |
| **15** | **Envelope Depth** | **0** |

The next table shows the settings for Oscillator 2, or DCO2. This offers the same controls, plus additional parameters for Cross Modulation and fine tuning. And there, in the Cross Mod options, is the clever bit… Sync1, which is hard sync of DCO2 (the slave) by DCO1 (the master).

The pitch of DCO2 is determined by parameters 21, 24 and 25, and these place it 14 semitones and 10 cents above DCO1. This means that, if not modulated by the LFO or an envelope, the slave will complete two-and-a-bit cycles for every master cycle, and its output will look much as shown in Figure 7. However, parameter 27 tells us that an envelope is modulating DCO2, so things are a little more complex than they might otherwise seem. No matter; let's move on…

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| **DCO2** | | |
| **21** | **Range** | **4'** |
| **22** | **Waveform** | **Sawtooth** |
| **23** | **Cross Modulation** | **Sync1** |
| **24** | **Tune** | **+2** |
| **25** | **Fine Tune** | **+10** |
| **26** | **LFO Depth** | **0** |
| **27** | **Envelope Depth** | **99** |

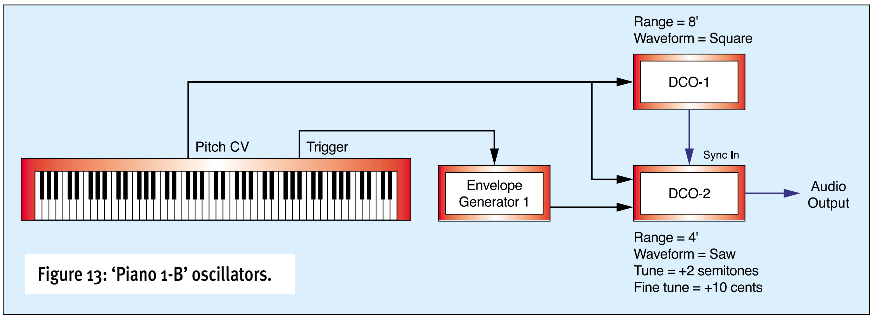


Figure 13: 'Piano 1-B' oscillators.

The third table contains just two parameters; the modulation settings for DCO1 and DCO2. The first determines whether keyboard velocity will affect the amount by which the Envelope Depth parameters (numbers 15 and 27) will affect the pitches of DCO1 and DCO2, but since it's set to 'Off' we can ignore it. The second parameter (number 32) determines which of the Tone's Envelope Generators will provide the pitch modulation whose amplitude is specified in parameters 15 and 27, and with what polarity. The '^1' setting means that Env1 is the modulation source, and with positive polarity.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| **DCO-MOD** | | |
| **31** | **Dynamics** | **OFF** |
| **32** | **Envelope Mode** | **^1** |

Now, as we saw a couple of paragraphs ago, the amount of pitch modulation applied to DCO1 is zero, but parameter 27 in DCO2 shows a maximum value of 99. This means that the pitch of DCO2 will be swept dramatically by Env1, which in turn means that there will be a huge variation in its sound as the contour progresses.

Given that both DCOs in each voice of a JX10 Tone track the keyboard, we can represent parameters 11 to 32 as shown in Figure 13 (above). This is identical to the classic 'swept' sync sound depicted in Figure 11, with the addition of keyboard tracking of DCO2.

The last table on this page, which shows the Mixer's parameters, tells us how the oscillators' outputs are mixed before being passed to the rest of the VCF/VCA signal path. Parameter 41 determines that, as well as driving the sync input of DCO2, the audio output of DCO1 is contributing 25 percent of its maximum amplitude to the mix. This means that it is performing two roles: one as the sync master, the other as a conventional audio source.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| **MIXER** | | |
| **41** | **DCO1** | **24** |
| **42** | **DCO2** | **99** |
| **43** | **Envelope Depth** | **99** |
| **44** | **Dynamics** | **1** |
| **45** | **Envelope Mode** | **^2** |

Parameter 42 suggests that the output from DCO2 is mixed at full amplitude (most JX10 parameters span a range from zero to 99) but this is not the whole story, because parameters 43, 44 and 45 also control the level of DCO2. The table shows that the positive polarity of EG2 is raising the level of DCO2 even further, according to the shape of its contour, and subject to a gain determined by playing velocity (the JX10 offers four keyboard response curves, programmed as Dynamics settings '1', '2', '3', or 'Off'). If all this sounds a bit of a jumble, don't worry, because it's much simpler to interpret as a block diagram, as shown in Figure 14 (below).

Adding Figure 14 to Figure 13 gives us

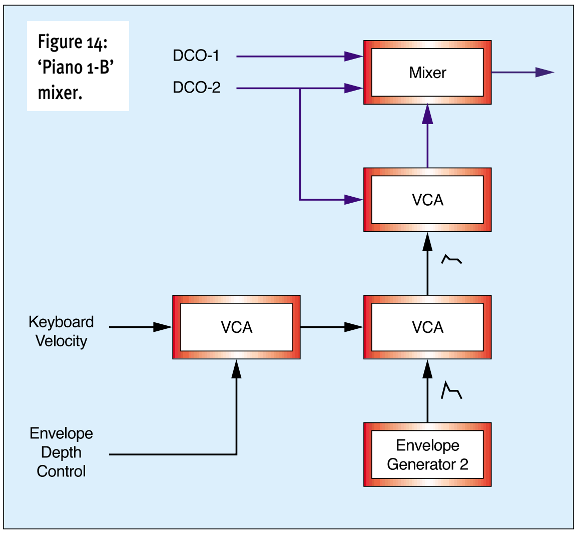


Figure 14: 'Piano 1-B' mixer.

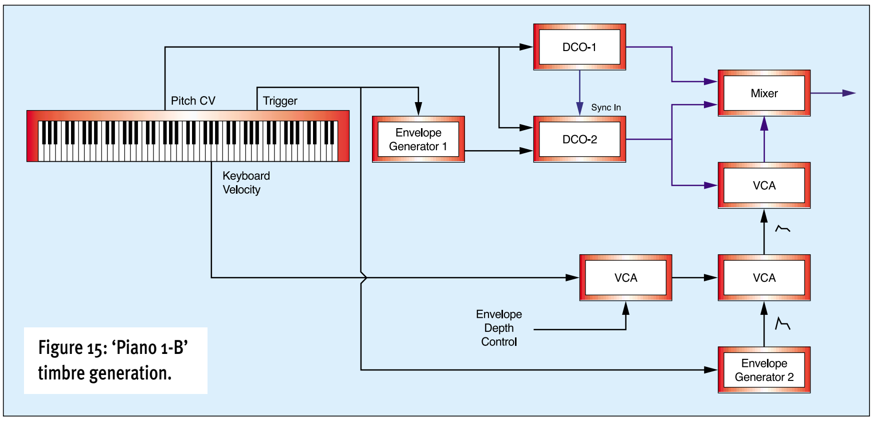


Figure 15: 'Piano 1-B' timbre generation.

Figure 15 (above), which shows how the JX10 produces the raw sound for the 'Piano 1-B' Performance.

Now we need to define the actions of two Envelope Generators. The tables below show these, while Figures 16 and 17 (right) show the contours themselves (on a JX10, setting Key Tracking to '1' means that the envelope times are halved from the bottom of the keyboard to the top, while setting Key Tracking to '2' means that they are quartered. This increasing of Decay and Release rates at higher pitches imitates the natural response of acoustic instruments).

## The Action Of Env1 & Env2

The tables below show that Env1 is modulating the pitch of DCO2 (the slave), so there's a rapid blip of sync sweep (and therefore a huge tonal blip) at the start of the note. This goes some way toward imitating the massive changes in harmonic structure that occur during the piano's hammer strike, and the exchange of energy among harmonics when the hammer bounces off the string. The initial phase of the sound is followed by a constant Sustain. Roland chose the Sustain Level such that the waveform generated by hard sync (with DCO2 offset by 14 semitones and 10 cents plus the Sustain Level) produces an appropriately 'wiry' tone.

Finally, once you release the key, Env1 travels through its Release phase, so the tuning offset of DCO2 falls to 14 semitones and 10 cents (the tuning interval defined in parameters 21, 24 and 25) generating a second, gentler sync sweep as it does so.

The mild tracking of approximately 20 percent is appropriate, because although these elements of the piano's sound become more rapid as you play further up the keyboard, they do not become dramatically so.

You'll remember that parameter 41 fixes the amplitude of DCO1 at approximately 25 percent. This means that we have an 8' square wave permanently present in the sound. At the same time, parameter 42 defines that DCO2 is permanently present at full amplitude.

However, both the final table on the previous page and Figure 14 show that Env2 is affecting the amount of sync'ed sound in the final output from the mixer. I doubt that DCO2 is sliding from 200 percent at the start of the note to 100 percent at the end, so I suspect that something like Figure 18 (right) is closer to the truth. Curiously, this means that the *un*-sync'ed sound from DCO1 becomes more important as the note progresses.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| ENV1 | | |
| 81 | A | 00 |
| 82 | D | 01 |
| 83 | S | 06 |
| 84 | R | 35 |
| 85 | Key Follow | 01 |

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| ENV2 | | |
| 91 | A | 00 |
| 92 | D | 70 |
| 93 | S | 00 |
| 94 | R | 40 |
| 95 | Key Follow | 02 |

## Putting It All Together

It would be nice to say that we have now plumbed the depths of the JX10's 'Piano 1-B' Performance but, inevitably, this is far from the whole story. If we were to leave the Tone in this state, it would drone on forever, as implied by the orange and purple arrows on the right of Figure 18. Pressing a note on the keyboard would merely produce blips of interest, as shown by the green section of the diagram. Clearly, we need to add more sound-shaping elements, and the JX10 provides these in the form of a conventional subtractive signal path. Unfortunately, we've run out of space for this month, so I'll complete the Tone next time, adding the VCF and VCA, and then demonstrating some other clever tricks that the JX10 has up its electronic sleeves. Until then...

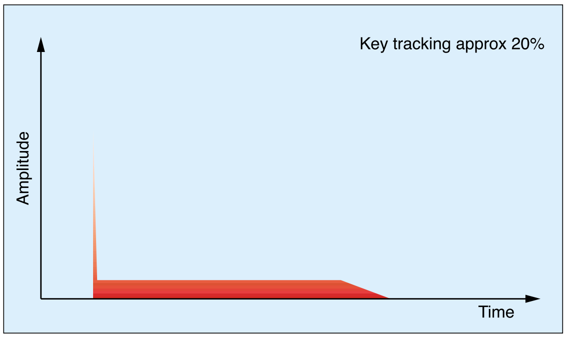


Figure 16: The contour generated by Env1 (affects the tone of the sync'ed sound).

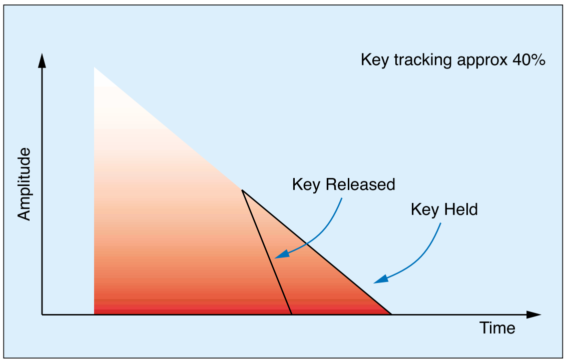


Figure 17: The contour generated by Env2 (affects the amount of sync'ed sound).

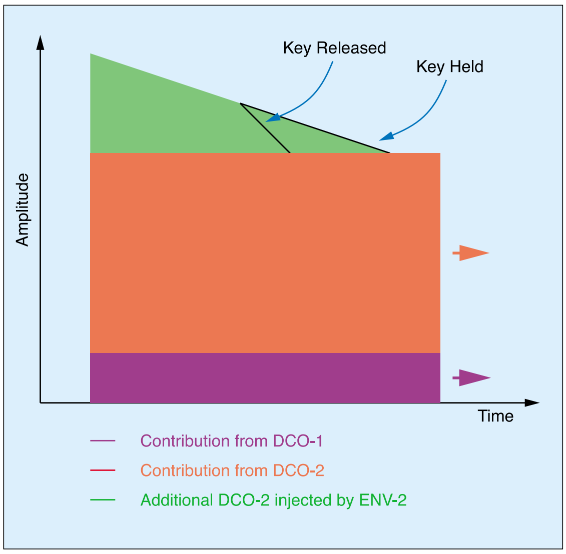


Figure 18: The relative contributions of the oscillators.

# Synth Secrets

## Synthesizing Acoustic Pianos On The Roland JX10

**How *did* they make that sound on a subtractive synth? We continue to dissect the analogue 'Acoustic Piano' Perfomance from Roland's 1986-vintage JX10.**

Gordon Reid

Last month, I discussed how we can use hard sync to generate radical changes in the harmonic structure of a sound. I then described how the JX10 uses its two contour generators to adjust the harmonic content and amplitudes of its two sync'ed oscillators to produce the basic timbre of an acoustic piano patch, the factory Tone entitled 'Piano 1-B'.

I have combined all of last month's parameter tables into the table on the right, and drawn the incomplete patch that they describe as Figure 1 (below). With these as our starting points, we can now take the output from the oscillator/mixer sections — shown as the blue arrow to the far right of the diagram — and mould it using the JX10's filter and VCA/chorus sections to create something that (I hope) will sound not too dissimilar to a genuine piano.

But, before going any further, let's pause to ask ourselves what it is that makes the piano such a special instrument. Well, the sound itself is remarkable, rich in harmonic and enharmonic components, and it evolves in a mind-bogglingly complex fashion at all stages throughout a note. But I want to look beyond this, and find what it is that makes playing the piano such a uniquely rewarding experience.

Let's start by eliminating a few possibilities. For example, the magic of the piano isn't found in the range of expression that you can coax from a single note. This is because, once you have pressed a key and the note has sounded, there's nothing you can do to control it other than to let it ring, or to curtail it by releasing the key. Instruments such as brass and woodwind — on which changes of blowing pressure and embouchure can create wide variations of tone and amplitude within a single note — are clearly superior in this area. Nor is the piano's special quality a consequence of the sheer power of its sound, nor its depth. A cathedral organ scores more highly in both these areas. It's not the sense of aggression you can obtain from the sound, because the overdriven guitar is (probably) tops in this area, and — at the other end of the spectrum — it's not because of a particularly sweet or soulful quality, because a solo violin or cello is far more emotive and evocative. It's not even

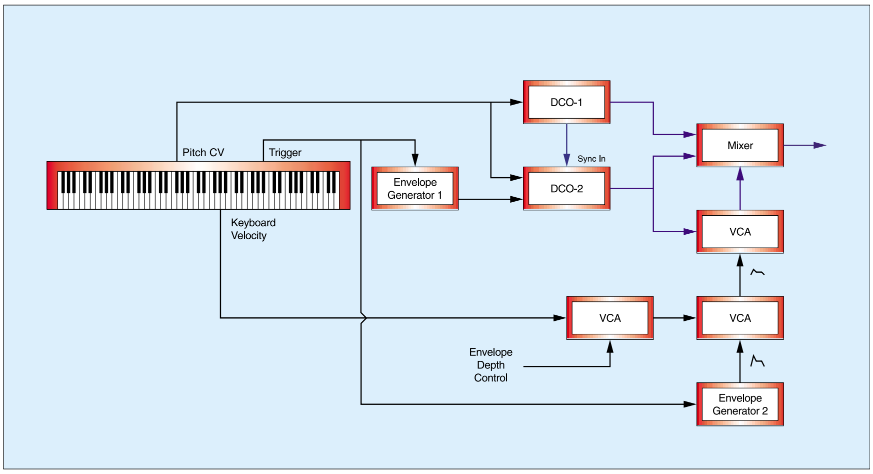


Figure 1: The parameters for the 'Piano 1-B' Tone in the table on the right represent this synth structure.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| DCO1 | | |
| 11 | Range | 8' |
| 12 | Waveform | Square |
| 13 | Tune | 0 |
| 14 | LFO Depth | 0 |
| 15 | Envelope Depth | 0 |
| DCO2 | | |
| 21 | Range | 4' |
| 22 | Waveform | Saw |
| 23 | Cross Modulation | Sync 1 |
| 24 | Tune | +2 |
| 25 | Fine Tune | +10 |
| 26 | LFO Depth | 0 |
| 27 | Envelope Depth | 99 |
| DCO-MOD | | |
| 31 | Dynamics | Off |
| 32 | Envelope Mode | ^1 |
| MIXER | | |
| 41 | DCO1 | 24 |
| 42 | DCO2 | 99 |
| 43 | Envelope Depth | 99 |
| 44 | Dynamics | 1 |
| 45 | Envelope Mode | ^2 |
| ENV1 | | |
| 81 | A | 00 |
| 82 | D | 01 |
| 83 | S | 06 |
| 84 | R | 35 |
| 85 | Key Follow | 01 |
| ENV2 | | |
| 91 | A | 00 |
| 92 | D | 70 |
| 93 | S | 00 |
| 94 | R | 40 |
| 95 | Key Follow | 02 |

a consequence of the piano's huge frequency range and polyphony because, although the aforementioned cathedral organ is the piano's equal in these areas, it nonetheless lacks the piano's *je ne sais quoi*. So what *is* the secret?

For me, the magic of the piano derives from its dynamic range. No other instrument is as flexible, and no other offers you such latitude to caress some notes while pounding away at others, mingling gentle melodies with pulsating rhythms and accompaniments. So, when we synthesize the piano, it's not enough to recreate an initial timbre and reproduce it polyphonically. In that direction lie the unquiet ghosts of unloved Crumars, Galantis, and Sound City Jo'annas. We need to do more... we need to make our patch respond to keyboard dynamics.

## Dynamics Explained

When referring to keyboard instruments, the terms dynamics and touch sensitivity refer to the velocity at which the key moves downward when you play it. Aftertouch, which is when you press a key more firmly into its bed after the initial impact, is a separate thing altogether.

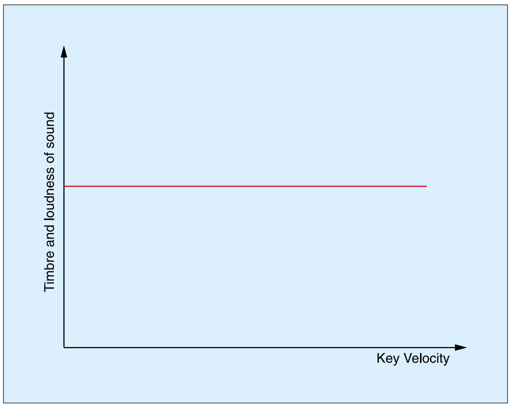
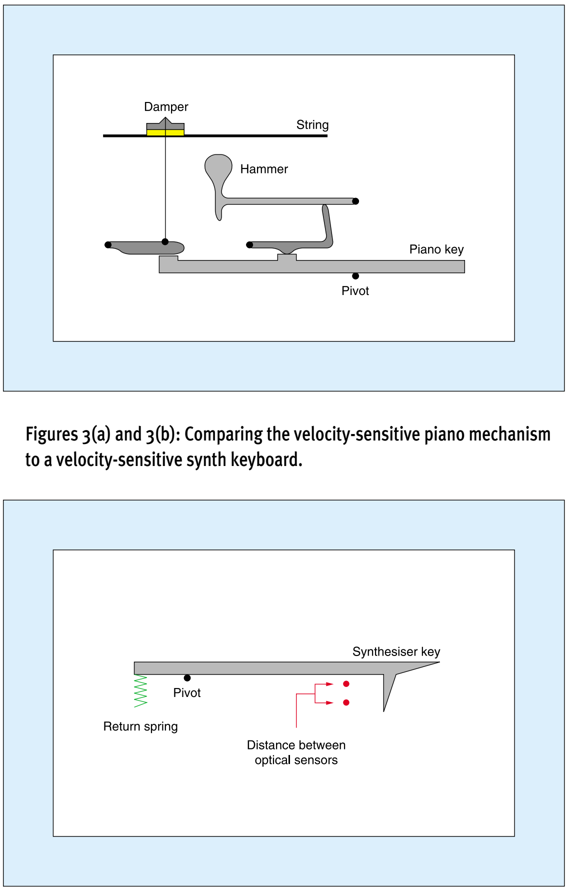


Figure 2: Increasing or decreasing the key velocity has no effect on the sound.

So let's start by considering the effect of playing a note 'softer' or 'harder' (ie. with lesser or greater velocity) on a powerful synth such as a Sequential Prophet 5, an Oberheim OBX, a Roland Jupiter 8 or a Moog Memorymoog. Despite their huge synthesis power, none of these offers velocity sensitivity, so, no matter how you play the key, the end result is always the same (this, of course, is one of reasons why early synths offered so many alternative controllers). We can represent this insensitivity as shown in Figure 2 above.

If we now recall the piano mechanism explained two months ago and shown again in Figure 3(a) above, it's clear that, when we depress a note more quickly, the various levers project the hammer towards the strings at a higher velocity, resulting in a greater amount of energy being transferred from the muscles in your fingers to the string. But a synth has no levers or strings... so we need to employ a different mechanism that generates the same effect.

One such mechanism might use two photocells that measure the length of time taken for the underside of the key to pass two points in its travel. If you consider the two red dots under the key in Figure 3(b), as shown above, it should be intuitively obvious that the velocity with which you depress the key is inversely proportional to the time that elapses between the key passing the first dot and then passing the second — in other words, the greater the velocity, the shorter the time interval. I have plotted this relationship in Figure 4, left.



Figures 3(a) and 3(b): Comparing the velocity-sensitive piano mechanism to a velocity-sensitive synth keyboard.

Now try to imagine how we could achieve on a synth the piano's effect of the note becoming louder the harder you hit a key. In fact, it's not so hard. If you map the velocity to the Gain of a VCA controlling the

▶

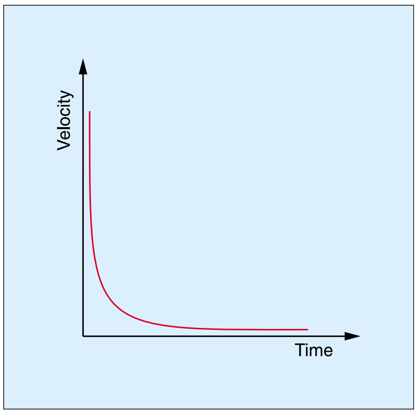


Figure 4: Plotting the relationship between key velocity and the time taken to travel the distance between the detectors.

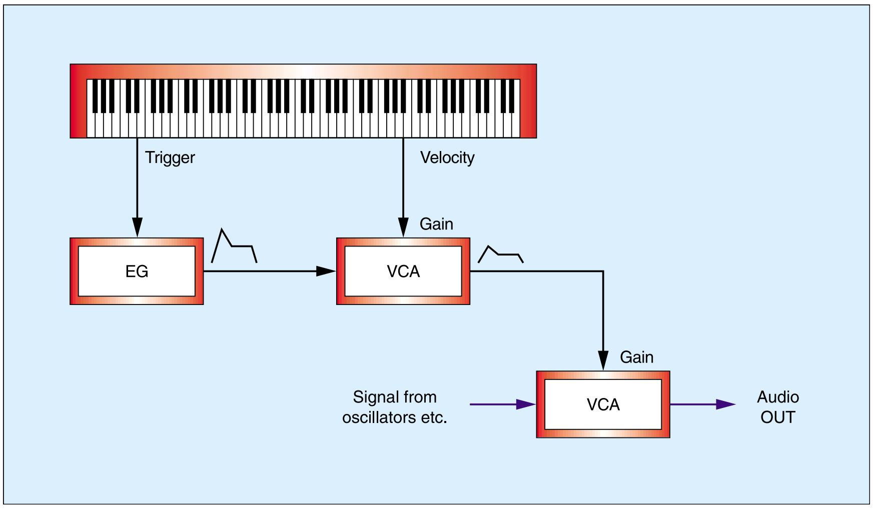


Figure 5: Using velocity to control the loudness of a sound.

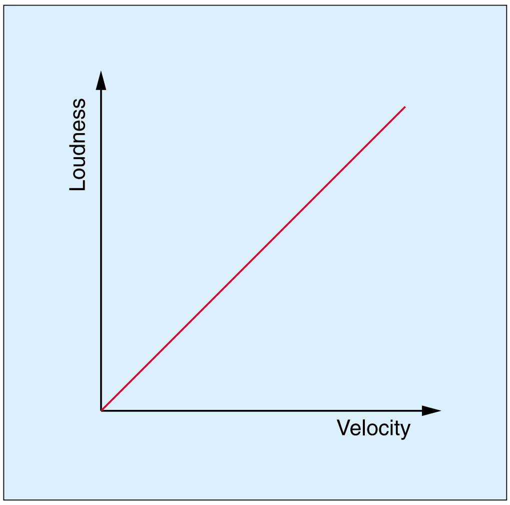


Figure 6: As the velocity increases, the sound gets louder — here, a linear relationship is shown.

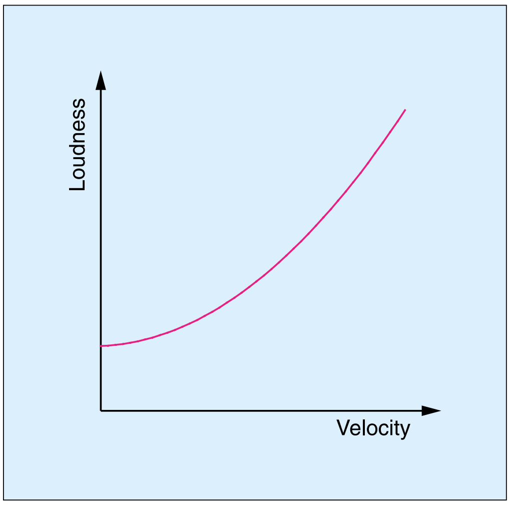


Figure 7: A non-linear relationship between velocity and loudness.

audio signal path (as shown in Figure 5 on the previous page) so that there is a proportional relationship between velocity and Gain, then the greater the velocity, the greater the Gain, and the louder the resultant note (see Figure 6 above). This is broadly consistent with the physical mechanism shown in Figure 3(a).

Of course, there's no reason why we should stick to simple proportional relationships such as those depicted in Figures 4 and 6. Modern synths are little more than specialised computers with application-specific I/O, so their designers can define any relationships they wish, by describing them mathematically. In this way, it's easy to replace the simple dynamics curve of Figure 6 with the non-linear one shown in Figure 7 (right).

Extending this idea still further, there's no reason why the parameter on the 'Y' axis of Figures 6 and 7 should be limited to Loudness. If we change this to 'amount', it could be the filter cutoff frequency, or the gain of a VCA controlling the LFO amount applied to the pitch of an oscillator... or indeed anything else in the synth. And therein lies the secret of 'dynamics'. By defining a suitable response to key velocity, and then directing it to appropriate modules within the voice structure, we can make the sound come alive in ways that are all but impossible without velocity sensitivity.

## Multiple Dynamics Responses In One Patch

Last month, when discussing the parameters in its Mixer section, I told you that the JX10 offers four dynamics curves of the type shown in Figures 4 and 6. Roland refers to these as '1', '2', '3' and 'Off', and I have drawn them in Figure 8 (below).

As you can see, 'Off' means that any signal controlled by Dynamics is insensitive to key velocity, as shown in Figure 2. This means that, if you set all the Dynamics parameters in the JX10 to 'Off' it emulates the aforementioned Prophet, OB, Jupiter or Memorymoog. Given the esteem in which these synths are held, this may seem like a good idea, but I suggest that it is not.

If you set the Dynamics to '1', a controlled signal responds in a way similar to that shown in Figure 7. At low velocities, the CV (or its digital equivalent) is still applied with an appreciable amount. Then, as you play with greater and greater velocity, the amount increases gradually. As you approach maximum velocity, the curve slopes sharply upwards until the CV is applied with 100 percent of its maximum amplitude.

Dynamics curve '2' is another

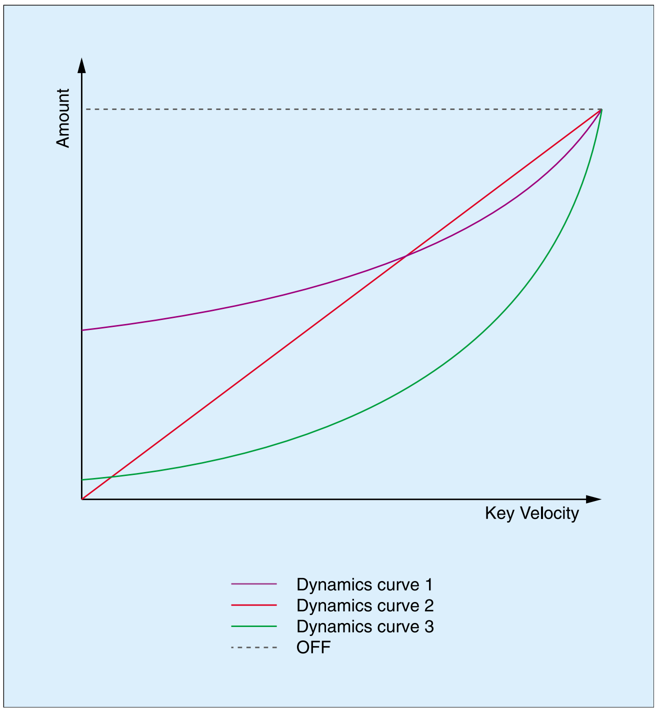


Figure 8: The four dynamics responses offered by the JX10.

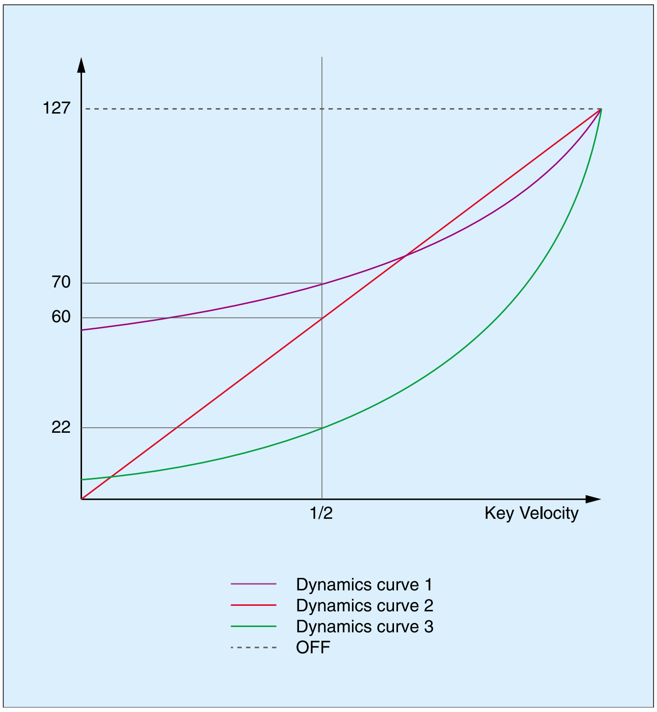


Figure 9: Generating different controller gains using different dynamics curves.

relationship we have already seen: the dynamics response is linear, without any offset. Finally, there's curve '3', which starts close to zero and then, after a fairly flat response for about half the dynamic range, curves sharply up to its maximum.

You may ask why we need four different responses. Well, imagine that you want to affect the signal level, the signal brightness and the amount of vibrato by different amounts, but all controlled by the key velocity. Figure 9 on the previous page shows that, by choosing different response curves, you could — with a single key velocity of, say, half the possible maximum — generate CVs of approximately 55 percent, 47 percent and 17 percent of maximum. Figure 10 (right) shows the block diagram for this, and demonstrates how you can make each element of the sound respond in a unique fashion.

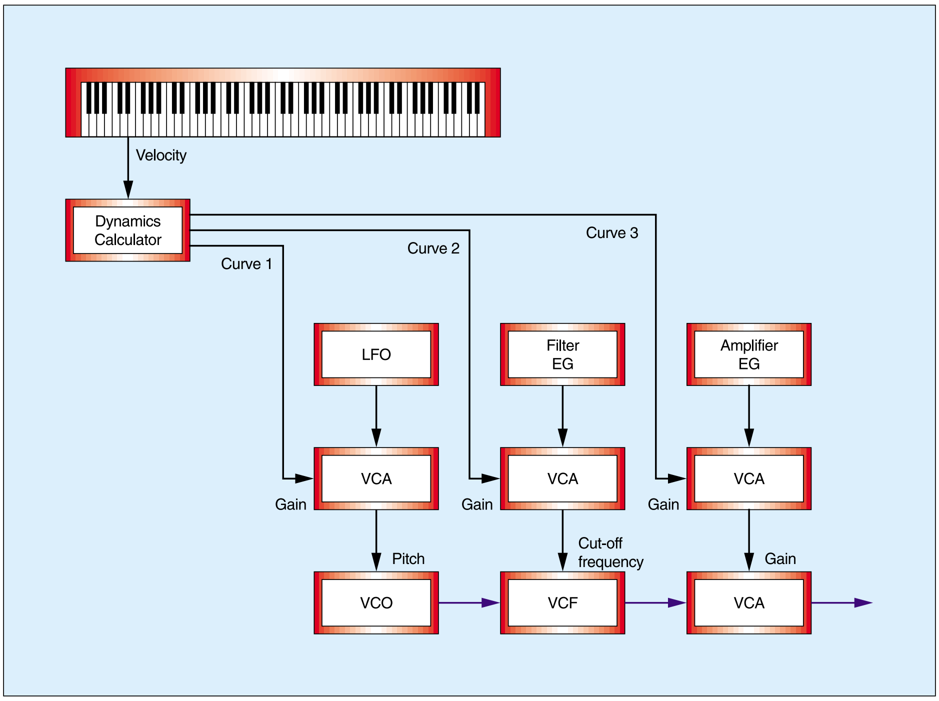


Figure 10: Obtaining different loudness, brightness and modulation responses from a single key velocity.

## The Brightness Response Of The 'Piano 1-B' Tone

So let's now return to the JX10's 'Piano 1-B' Tone, and see how Roland's voice programmers took last month's sync'ed sound and shaped it to create a patch that sounded similar to, and responded in a similar way to, an acoustic piano.

The table on the right shows the parameters contained in the JX10's powerful filter section, together with the values used in the patch. We can represent this as shown in Figure 11 (below).

The parameters show that the high-pass filter (HPF) is inactive; all the low frequencies in the sound are allowed to pass unhindered. In contrast, the unmodified cutoff frequency of the low-pass filter (LPF) is set to a value of '53' (out of a maximum of 99), suggesting that it lies close to the middle of its range. The resonance of '2' is tiny... barely perceptible, but perhaps enough to create a slight accent to the sound. The Key Follow of '24' determines that the cutoff frequency rises as you play up the keyboard, but only at 24 percent of the rate of increase in the pitch of the

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| VCF | | |
| 51 | HPF | 0 |
| 52 | LPF Frequency | 53 |
| 53 | LPF Resonance | 02 |
| 54 | LFO Amount | 0 |
| 55 | ENV Amount | 14 |
| 56 | Key Follow | 24 |
| 57 | Dynamics | 1 |
| 58 | Envelope Mode | ^2 |

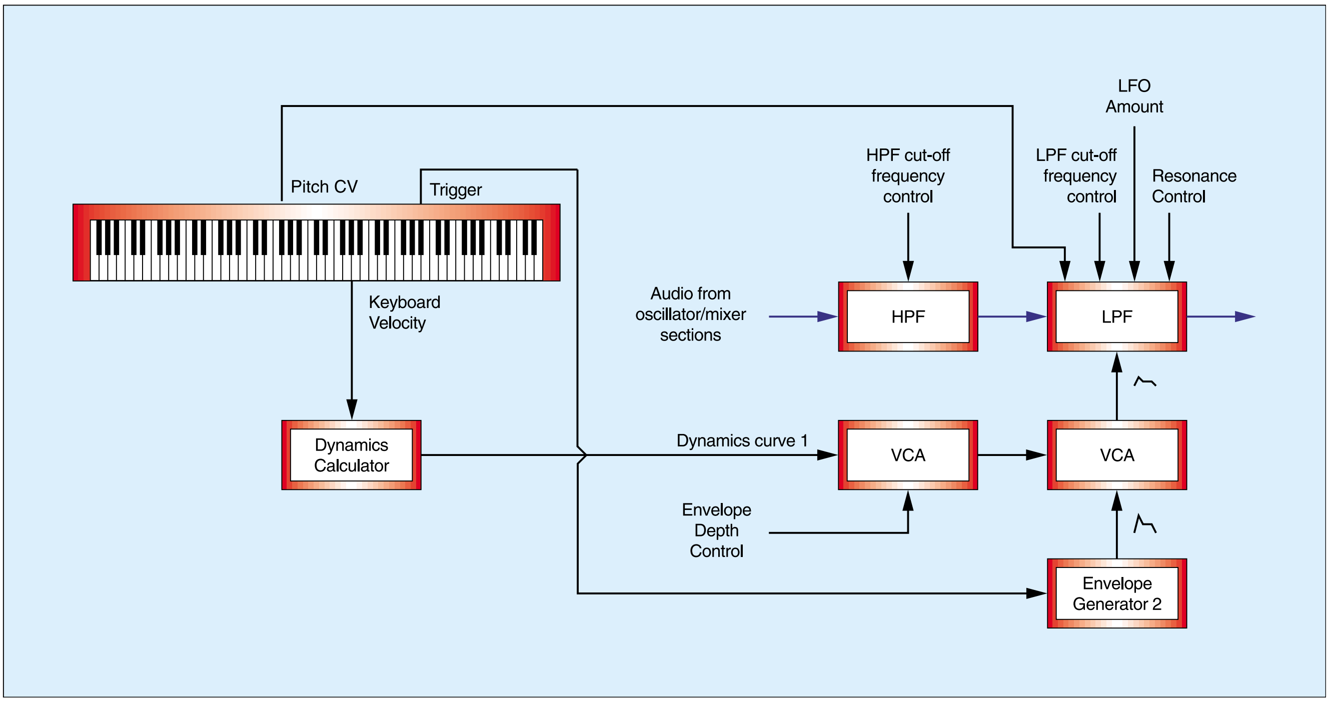


Figure 11: The filter block diagram for the 'Piano 1-B' Tone.

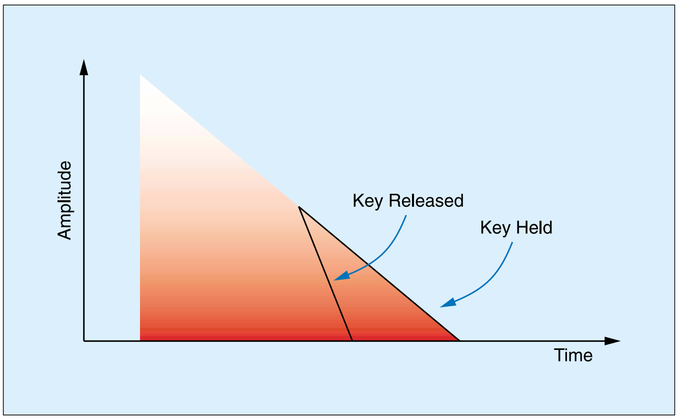


Figure 5: Using velocity to control the loudness of a sound.

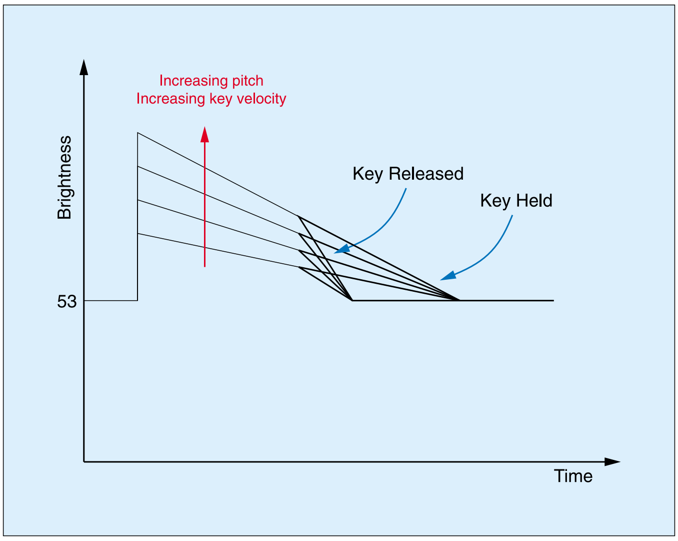


Figure 13: The brightness response of the 'Piano 1-B' Tone.

sound. This means that — all other things being equal — higher notes are brighter than lower ones, but in a relatively subtle way.

The LFO does not affect the cutoff frequency, but the second envelope does so (parameter 58) at a maximum amplitude (controlled by Dynamics curve 1) of approximately 14 percent of its maximum (as defined by parameter 55). We know the shape of the second envelope (parameters 91 to 95) because we used it last month. Shown again in Figure 12 (top), we can see that it is a fair representation of the shape of a piano note.

Knowing all of the above, we can now draw a diagram (Figure 13, above) that represents the filter response. Note that the value of 53 in parameter 52 ensures that the cutoff frequency can never fall to zero, so the note is never fully shut off by the filter.

## The Amplitude Response Of The 'Piano 1-B' Tone

Having defined the filter response, we can now determine the final stage in the 'Piano 1-B' signal path; the output amplifier and its associated chorus unit. There are four parameters for this, as shown in the table on the right.

These settings show that the maximum level is approximately 96 percent, that the VCA contour is determined by the second envelope, and that the audio signal gain responds to dynamics according to curve 2 in Figure 8. The choice of dynamics curve 2 is an interesting one, because it means that, at low velocities, the sound is produced at vanishingly low amplitudes. This is not quite representative of a real piano, in which the hammer does not *just* hit the string; if you press the key fast enough for the hammer to reach the string it will be travelling fast enough to produce a clearly audible note. I might be tempted, therefore, to change this to curve 3, which has the correct response at low velocities, but which would require a much heavier touch to obtain middling amplitudes.

The final parameter in this group defines the status of the chorus unit. Since chorus would fatten the sound in a way that is

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| VCA/CHORUS | | |
| 61 | Level | 95 |
| 62 | Env Mode | E2 |
| 63 | Dynamics | 2 |
| 64 | Chorus | Off |

inappropriate for an acoustic piano simulation, this is set to 'Off'. We can therefore draw the VCA/Chorus parameters as shown in Figure 14 (below).

## Modulation (Or The Lack Of It)

The last table this month (below) includes the final set of parameters in every JX10 patch; those that define the LFO modulation generator. Although these contain values (because, on the JX10, they must) they are irrelevant. This is because the LFO has no effect at any of the points in the patch where it could modulate the pitch or brightness. In other words, the values for parameters 14 and 26 (pitch modulation of DCO1 and DCO2 respectively) and 53 (VCF cutoff frequency modulation) are '0'.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| LFO | | |
| 71 | Waveform | Rand |
| 72 | Delay | 0 |
| 73 | Rate | 76 |

## 'Piano 1-B'

So now it's time to put the whole patch together. This is a non-trivial task, and one for which the *SOS* graphics department are going to pin my photo to the nearest dartboard. Nonetheless, it's important that we should see how the various parts of the patch interact. Here goes... take a look at Figure 15 on the next page. I have simplified this diagram slightly to make it manageable, and, as in previous months, I have omitted a bunch of mixers and multiples to improve legibility. Nonetheless, it's still a daunting diagram, and this reflects the complexity of

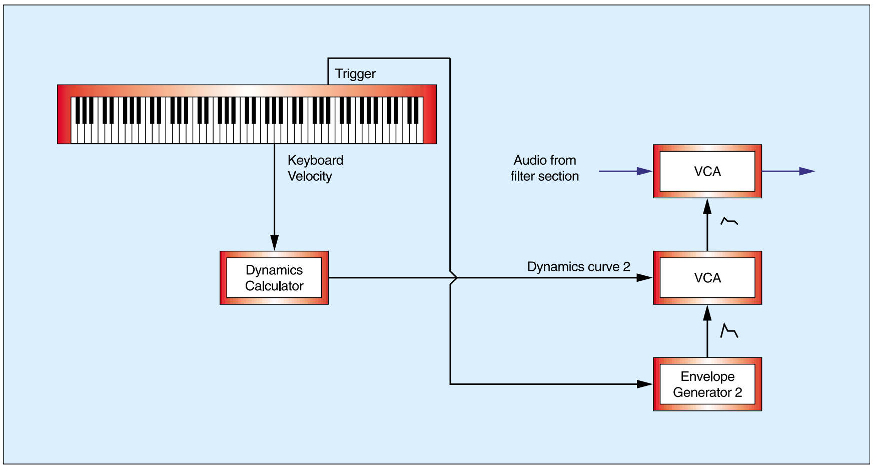


Figure 14: The amplifier/chorus section of the Tone.

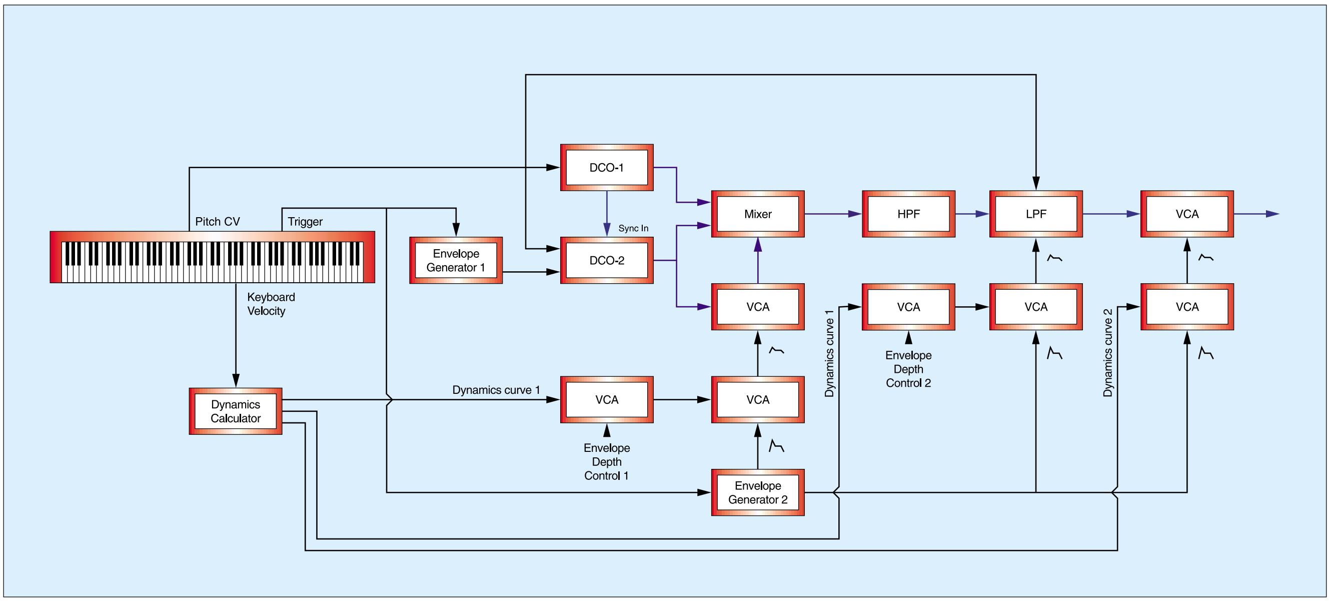


Figure 15: The 'Piano 1-B' Tone block diagram.

the JX8P and Super JX10 voice structure. Oh, and don't forget... this represents just *one* of the 12 voices in a JX10, so with the exception of the keyboard, every block must be repeated 12 times (see Figure 16, below).

Despite this, the signal path for each voice remains relatively simple, and is not too different from that on basic synths such as the Roland SH101, ARP Axxe, Korg MS20 and Minimoog that we have considered over the past year or so. Look at it... it's just a couple of oscillators passed through a high-pass filter, a low-pass filter and an amplifier. Clearly, the complexity lies in the JX10's ability to *control* these components using its assignable contour generators and touch-sensitivity. And that, dear reader, is one of the great secrets of synthesis: as is so often the case, it's not what you've got, it's what you can do with it that matters.

Indeed, we can apply this adage to almost every area of synthesis. You can be programming string sounds, brass, percussion, or a wide variety of different sounds real or imagined, and the attention you pay to dynamic response will almost certainly repay itself many times over.

## And The Result?

So, how does 'Piano 1-B' sound? Well, to my ears it's a wonderful imitation of an electro-mechanical piano. It has some of the timbre of my beloved RMI Electrapiano but, with dynamics, it is of course much more flexible than this. On the other hand, it's not exactly a Fender Rhodes 73 or a Wurlitzer EP200 either, although it occupies the same sonic territory as both of these.

However, 'Piano 1-B' is *not* similar to an acoustic grand piano. Each note responds in a piano-like fashion, and with the JX10 in 'Whole" mode (in which one patch utilises all 12 voices) there is sufficient polyphony to play it in piano-like fashion. But it still doesn't *sound* like a piano.

This means that, despite my promises of the past couple of months, I haven't fulfilled my promise to create an acoustic piano patch on a subtractive synth... there's still something missing. So, next month, we'll continue to press forward on this noble quest, and discover yet more tricks that the JX10 has tucked away in its analogue electronic sleeves. Until then... SOS

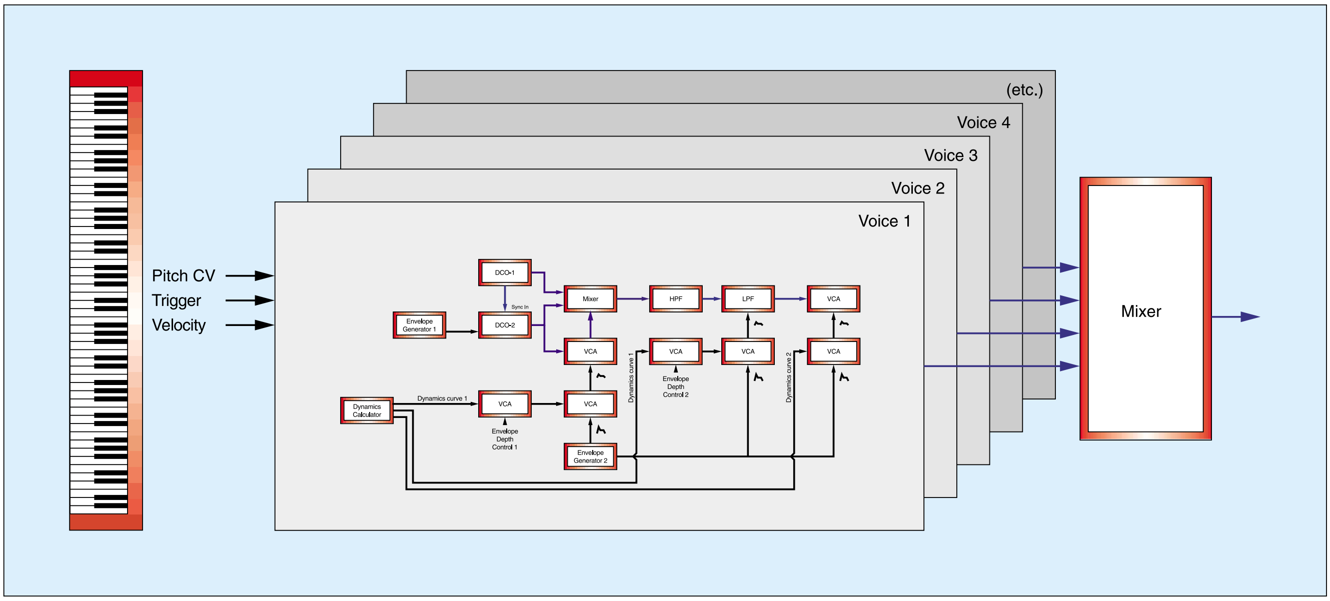


Figure 16: A polyphonic representation of the 'Piano 1-B' Tone.

# Synth Secrets

## Synthesizing Acoustic Pianos On The Roland JX10

Gordon Reid

**When trying to copy a real piano with an analogue synth, if one patch doesn't quite do it, two just might...**

For the past three instalments of Synth Secrets, I've been discussing the nature of the piano and looking at the ways in which we can attempt to recreate its sound. But even after all this, the best I have yet been able to manage is something that sounds similar to an electro-mechanical piano. (Of course, synthesizing the Fender Rhodes or Wurlitzer EP200 is no bad thing...) Numerous analogue pianos were released between 1970 and 1985, peaking with the superb Roland MKS10 rackmount module. But even this survived just two years before the introduction of samploid synths, and Roland's own 'SAS' piano synthesis swept the analogue piano genre away as if it had never existed.

The demise of analogue piano synthesis is, in some ways, a shame. Although it never achieved the authenticity that early synth programmers had anticipated, it led to the creation of a family of new, piano-like sounds, the best of which exuded a character of their own, and which have now been all-but lost. So, to conclude this discussion of piano synthesis using analogue, subtractive techniques, I'll finish describing the Roland Super JX10 performance that I used as a stage piano in 1986 and 1987, prior to purchasing the first of a pair of SAS-based Roland MKS20s that I still use today.

| PARAMETER NO. | PARAMETER | PIANO 1B | PIANO 1A |
|---|---|---|---|
| DCO1 | | | |
| 11 | Range | 8' | 8' |
| 12 | Waveform | Square | Sawtooth |
| 13 | Tune | 0 | 0 |
| 14 | LFO Depth | 0 | 0 |
| 15 | Envelope Depth | 0 | 0 |
| DCO2 | | | |
| 21 | Range | 4' | 2' |
| 22 | Waveform | Sawtooth | Pulse |
| 23 | Cross Modulation | SNC (Sync) 1 | SNC (Sync) 2 |
| 24 | Tune | +2 | +9 |
| 25 | Fine Tune | +10 | -07 |
| 26 | LFO Depth | 0 | 0 |
| 27 | Envelope Depth | 99 | 0 |
| DCO-MOD | | | |
| 31 | Dynamics | OFF | OFF |
| 32 | Envelope Mode | ^1 | v1 |
| MIXER | | | |
| 41 | DCO1 | 24 | 99 |
| 42 | DCO2 | 99 | 44 |
| 43 | Envelope Depth | 99 | 45 |
| 44 | Dynamics | 1 | 1 |
| 45 | Envelope Mode | ^2 | ^1 |
| VCF | | | |
| 51 | HPF | 0 | 0 |
| 52 | LPF Frequency | 53 | 52 |
| 53 | LPF Resonance | 02 | 01 |
| VCF continued | | | |
| 54 | LFO Amount | 0 | 0 |
| 55 | ENV Amount | 14 | 19 |
| 56 | Key Follow | 24 | 17 |
| 57 | Dynamics | 1 | 1 |
| 58 | Envelope Mode | ^2 | ^1 |
| VCA/CHORUS | Parameter | Value | |
| 61 | Level | 95 | 92 |
| 62 | Env Mode | E2 | E2 |
| 63 | Dynamics | 2 | 2 |
| 64 | Chorus | OFF | OFF |
| LFO | | | |
| 71 | Waveform | Rand | Sine |
| 72 | Delay | 0 | 0 |
| 73 | Rate | 76 | 80 |
| ENV1 | | | |
| 81 | A | 00 | 00 |
| 82 | D | 01 | 65 |
| 83 | S | 06 | 35 |
| 84 | R | 35 | 45 |
| 85 | Key Follow | 01 | 01 |
| ENV2 | | | |
| 91 | A | 00 | 00 |
| 92 | D | 70 | 60 |
| 93 | S | 00 | 00 |
| 94 | R | 40 | 36 |
| 95 | Key Follow | 02 | 01 |

The combined parameter table for Piano 1B and Piano 1A.

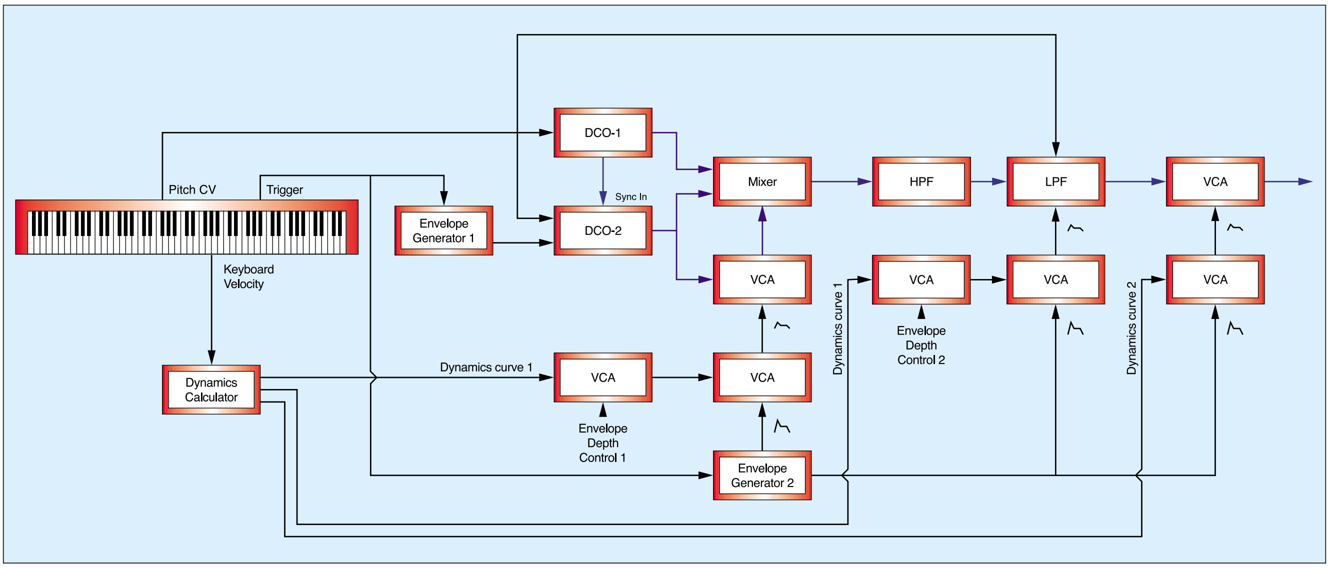


Figure 1: The Piano 1B block diagram.

## A Second JX10 Piano

The table below left again shows the Piano 1B patch with which I concluded last month's Synth Secrets, and Figure 1, above, shows the architecture that this describes. However, as you can see, the table also includes the values for another JX10 electric piano patch that — for reasons that will soon become clear — is called Piano 1A.

Superficially, the columns for Piano 1A and Piano 1B might look similar, but this is misleading. It's a bit like saying that all Minimoog patches must sound similar because a photograph of the same control panel patched to produce the sound of a piccolo looks pretty much the same as a photograph of the control panel set up to produce a rumbling bass. In other words, the JX10 has an architecture which, when represented in table form, always looks the same. But this too is misleading. The JX10's architecture is not entirely fixed; some parameters allow you to alter the way in which its sections interact with one another. If you think that this sounds suspiciously like a description of a modular synth, you are — to some extent — correct. Although the degree of flexibility involved is a fraction of that offered by a true modular, the Super JX10, like most powerful synthesizers, allows you to 'patch' certain elements to create different architectures.

### Nomenclature

**Please note that throughout this article I shall use the conventional term 'Patch' to refer to what Roland calls a Super JX10 'Tone', and 'Performance' to refer to what Roland calls a Super JX10 'Patch'. I could stick to the company's usage, but I suspect that this would be more confusing for everybody.**

To see how this works, let's consider parameters 23, 32, 45, 58, and 62. The first of these, parameter 23, 'Cross Modulation', allows you to patch the oscillators in three quite different ways. As we discussed two months ago, SNC1 is hard synchronisation of DCO2 (the slave) by DCO1 (the master). In contrast, XMOD is frequency modulation of DCO2 (now acting as the carrier) by DCO1 (the modulator). The third option, named SNC2, is hard synchronisation of DCO2 (the slave) by DCO1 (the master) where DCO2 is also acting as the FM carrier for DCO1 as modulator. As you would expect, SNC1 and XMOD create very different sounds but, because the effect of hard sync is the dominant factor in SNC2, this option sounds similar to SNC1, if somewhat richer in the mid and high frequencies.

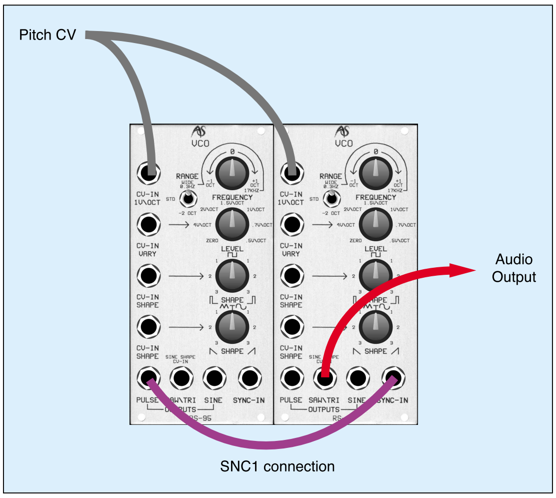


Figure 2: Two oscillators linked to produce hard sync (JX10 option: SNC1).

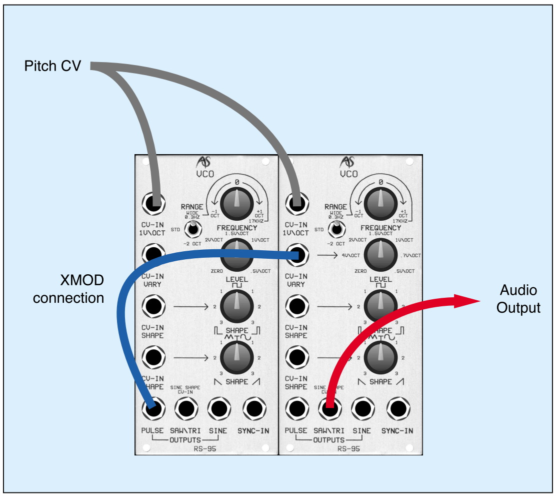


Figure 3: The same two oscillators linked as an FM pair (JX10 option: XMOD).

Got all that? No? Well, maybe figures 2, 3 and 4 (next page) will help, because these illustrate the same options using two patchable analogue oscillators. A picture may not always be worth a thousand words, but in this case, three pictures are worth a few hundred.

The remaining four parameters from the list above (numbers 32, 45, 58 and 62) are all Envelope Mode selectors that allow you to determine which envelope generator affects the pitches of DCO1 and DCO2, the contribution of DCO2 to the mix, the LPF cut-off frequency, and the VCA Gain (respectively) and with which polarity they do so. This is a far cry from the facilities offered by a true modular synth, but it still extends the range of sounds that the JX10 can produce. So, having

▶

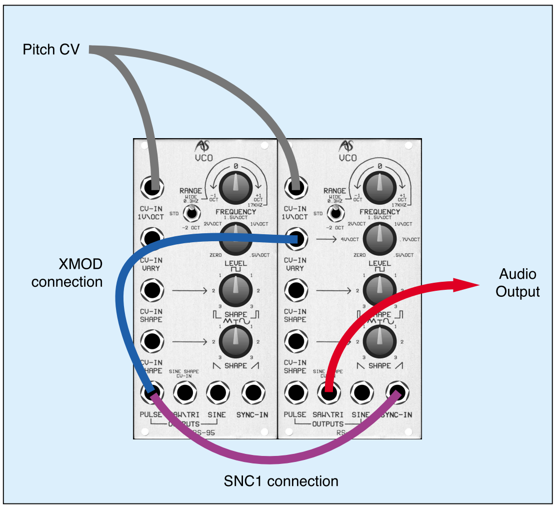


Figure 4: The two oscillators linked as an FM pair and as a sync'd pair (JX10 option: SNC2)

understood all of the above, let's now inspect the differences between Piano 1A and Piano 1B.

## The Piano 1A Oscillators

Starting with the oscillators, the relationships between DCO1 and DCO2 are quite different in the two patches. Whereas last month's patch used a square-wave master and a sawtooth slave, this month's starts with a sawtooth master and a pulse-wave slave. In previous Synth Secrets, I have stated that, when hard sync'd, the waveform of the master oscillator should make no difference to the sound produced. On the other hand, the shape of the slave is extremely relevant to the output, because it changes the harmonic content of the resulting waveform. (See Figures 5a and 5b, below.) As is intuitively obvious from these figures, the tones of these waves will be quite different from one another.

So is the shape of DCO1 irrelevant? No. Because Piano 1A uses SNC2, DCO1 and DCO2 are also acting as a pair of FM operators. This means that the waveform of DCO1 will have an effect on the output of DCO2. To be honest, this effect can be somewhat subtle, but when you are programming sounds deterministically (rather than using blind serendipity in the hope that you might stumble across something pleasing) it can be the difference between an acceptable patch and a superb one.

Even more significant is the change of the pitch relationship between DCO1 and DCO2. Piano 1B had an offset of a little over 14 semitones. Piano 1A has an offset of a little under 33 semitones. This makes a huge difference to the output waveform and its harmonic content. What's more, whereas the pitch of the slave in Piano 1B is swept by ENV1 (parameters 27 and 32) the frequency relationship of DCO1 to DCO2 in Piano 1A is constant throughout the note. This is because the value of parameter 27 is zero, thus making parameter 32 irrelevant.

Hang on... if there is no sync sweep at the start of the sound, does this render redundant the last two months' discussion of sync and its importance to the attack of the piano sound? It seems to. The use of both hard sync and FM in SNC2 is creating a complex new waveform but, unlike in the case of Piano 1B, the 'Cross Modulation' in Piano 1A is not imparting any blip to the front of the sound. You can hear this (or, rather, the lack of it) if you play the two patches one after the other. The first few milliseconds of Piano 1B exhibit a definite clunk, especially in the middle and lower octaves. Piano 1A lacks this and, as a result, its attack is less defined.

Moving on, you can see that Piano 1A's DCO1 is contributing its full amplitude in the Mixer, whereas DCO2 is contributing just 44 percent of maximum, plus an amount shaped by ENV1. Having discussed the relevant issues in depth over the past couple of months, I'll leave it to you to work out the effects of the ADSR, Key Follow (parameter 85), and Dynamics (parameter 44). Why should I do all the work?

## Digital Parameter Access

**If you're a regular reader of Synth Secrets, it can't have escaped your notice that I've used the last two parts as a bit of a tutorial on understanding Digital Parameter Access programming systems. In the past, these have attracted a great deal of criticism, almost to the point of hysteria, and no doubt some analogue anoraks will continue to heap opprobrium upon synths that use DPA. But I hope that I have shown that this is simply a different way to represent the same sound-making facilities that you will find on the knobbiest of analogue synthesizers, and to control the sounds thus produced.**

**Certainly, the Roland PG800 programmer, with its knobs and sliders, makes it altogether easier to program the JX10, but even this controls fewer than half of the parameters offered by the instrument. You might wish for things to be otherwise, and for all synths to be festooned with dedicated knobs and sliders. But when you consider that the JX10's DPA tables contain 147 parameters (that's 44 for each patch, and 59 for the performance and MIDI system) — and that's not including the synth's physical performance controls, nor the parameters that control these controls, nor the 'Chase Play', nor the fledgling sequencer — you'll soon realise that it ain't gonna happen.**

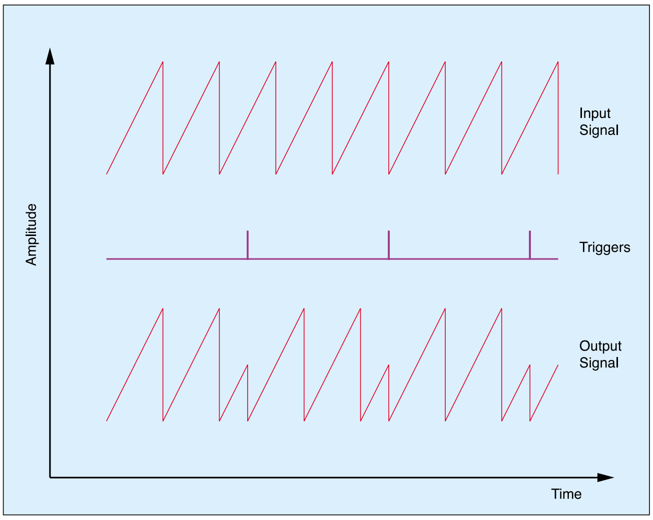


Figure 5a: A sawtooth slave of frequency F sync'd by a master with frequency 0.4F.

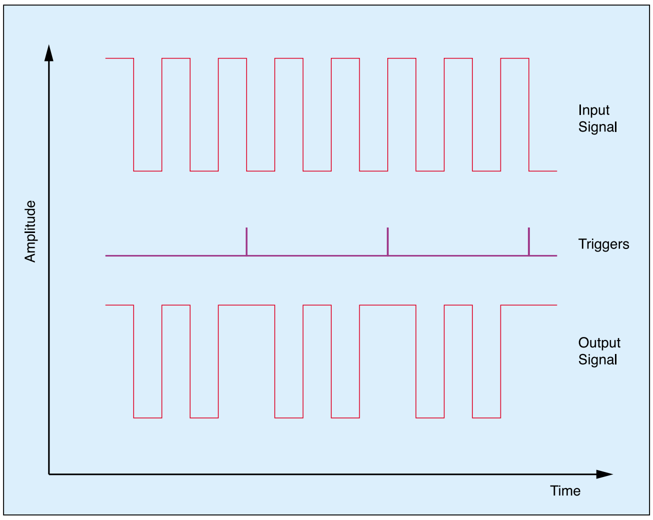
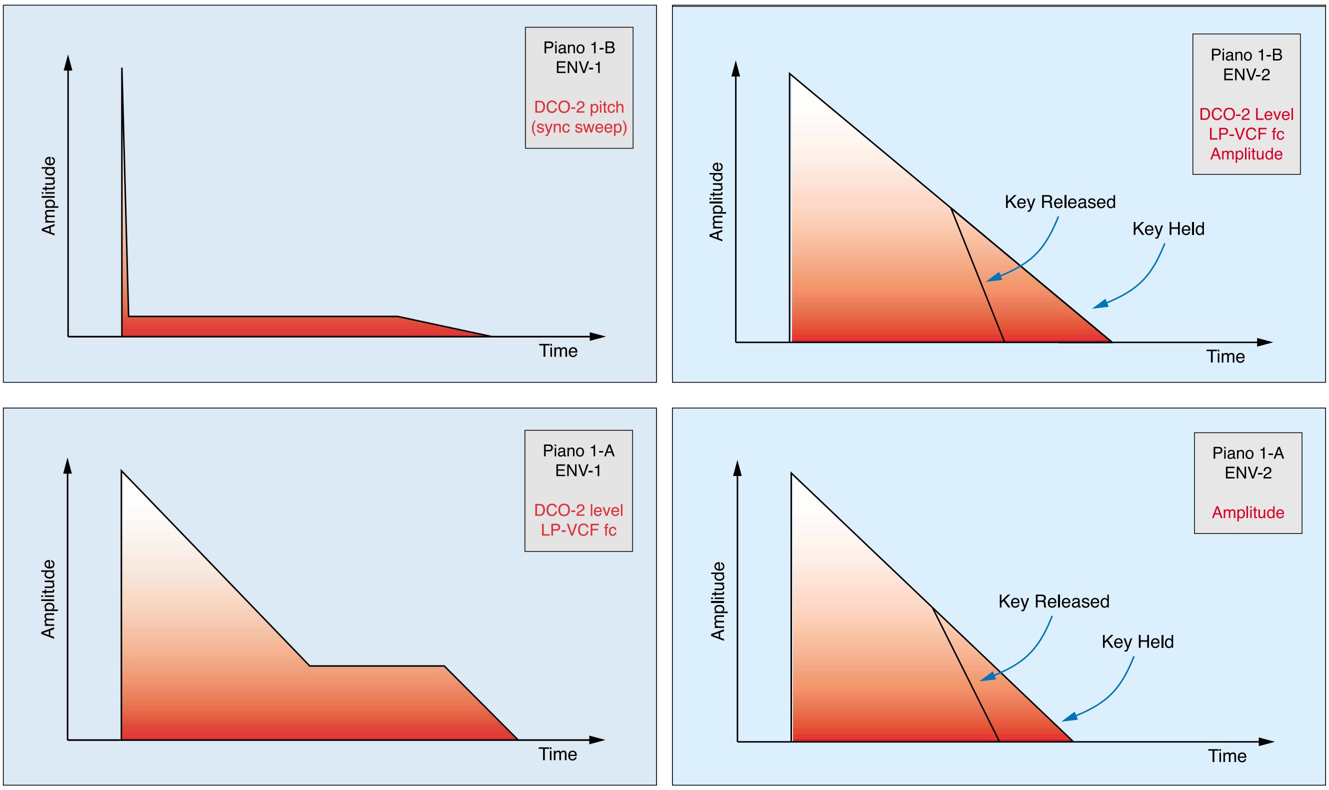


Figure 5b: A square wave slave of frequency F sync'd by a master with frequency 0.4F



Figures 6a-6d: The four ADSR envelopes used in Piano 1A and Piano 1B.

## Filters, Amplifiers And Envelopes

Looking at the rest of the table on page 136, we can see that there is a great deal of similarity between Piano 1A and Piano 1B. The filter settings are similar, the VCA/Chorus is almost identical, and the LFO remains irrelevant.

The greatest difference lies in the envelope shapes, and the patching of them.

Figures 6a to 6d, above, represent the ENV1 and ENV2 contours for each patch, and show the assignment for each.

At first sight, these seem quite similar, but the only common shape/destination is that of the two ENV2s, which control the total amplitude of their respective sounds. This means that the sweep of the filter and the contribution of DCO2 are quite different in each case.

To conclude this analysis of Piano 1A, I'll draw your attention to the block diagram equivalent to Figure 1. (See Figure 7, below.)

If you compare this to Figure 1, you can see the differences discussed above; the additional FM connection between DCO1 and DCO2, and the altered assignments for ENV1 and ENV2.

As I did last month, I'm now going to ask: how does it sound? Well, there's the lack of the clunk, which disappeared when the sync sweep was removed from Piano 1A. But a more significant difference is that Piano 1A is brighter, with more body in the mid frequencies. Overall, it sounds like a good 'analogue' piano patch, but one that makes little attempt to recreate the nuances of a real piano, or even a real electric piano

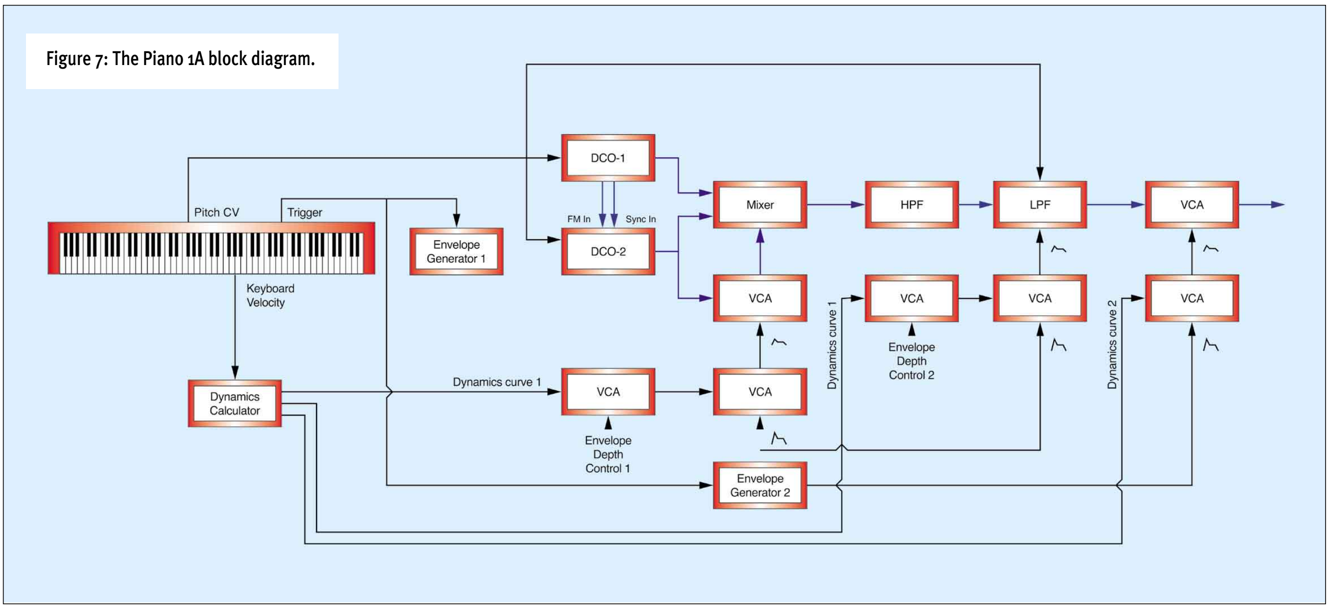


Figure 7: The Piano 1A block diagram.

| PARAMETER NO. | PARAMETER | VALUE |
|---|---|---|
| SYSTEM | | |
| 11 | Upper/Lower Balance | 50 |
| 12 | Dual Detune | +13 |
| 13 | Upper Split Point | |
| 14 | Lower Split Point | |
| 15 | Portamento Time | |
| 16 | Bend Range | 2 |
| 17 | Key Mode | Dual |
| 18 | Total Volume | 82 |
| AFTERTOUCH | | |
| 21 | Vibrato | 0 |
| 22 | Brilliance | 0 |
| 23 | Volume | 0 |
| UPPER MODULE | | |
| 31 | Tone Number | 22 (Piano 1B) |
| 32 | Chromatic Shift | -12 |
| 33 | Key Assign | Poly 1 |
| 34 | Unison Detune | |
| 35 | Hold | On |
| 36 | LFO Mod Depth | 01 |
| 37 | Portamento On/Off | OFF |
| 38 | Bend On/Off | On |
| LOWER MODULE | | |
| 41 | Tone Number | 21 (Piano 1A) |
| 42 | Chromatic Shift | -12 |
| 43 | Key Assign | Poly 1 |
| 44 | Unison Detune | |
| 45 | Hold | On |
| 46 | LFO Mod Depth | 0 |
| 47 | Portamento On/Off | Off |
| 48 | Bend On/Off | On |

Layering Piano 1A and Piano 1B into 'H1: Acoustic Piano'.

such as a Wurlitzer or Rhodes. So what use is it?

## An Introduction To Layering

In isolation, neither Piano 1A nor Piano 1B have a great deal to recommend them. Sure, they're usable in a '1985' sort of way, but they offer little that makes them cry out "Use Me". Fortunately, the Super JX10 is not just the 12-voice analogue synthesizer that we have been considering for the past three months. It is also two independent six-voice synthesizers.

You control the two halves of the JX10 using a second table of parameter values, divided into Patch (which we would normally call 'Performance') and MIDI. Called the Edit Map, this table offers no fewer than 59 parameters, and it is larger than the patch table used to edit the patches themselves. I have, therefore, confined the next part of this discussion to the parameters used to layer two patches into a single, composite sound.

The table on the left shows the parameters and values used in the Roland factory Performance 'H1: Acoustic Piano' which, as you might already have guessed, comprises Piano 1A and Piano 1B.

Starting with the System parameters, the first to consider is number 17, which states that the JX10 is in Dual mode, meaning that the two patches are layered one upon the other across the entire width of the keyboard. This, for reasons that I hope are obvious, makes parameters 13 and 14 irrelevant. Because portamento is Off in parameters 37 and 47, the portamento value is also irrelevant, and there is no slew between notes. However, for some unfathomable reason, Roland saw fit to program a pitch-bend range of two semitones for this Performance — not just weird, actually wrong. This then leaves the balance between the Upper and Lower patches, which is set to 50/50, and the detune between them. The detune value of +13 (on some arbitrary Roland scale) is a subtle difference, but proves to be important, and we shall return to this later.

The next bunch of parameters refers to aftertouch and these, as they must be, are set to zero. Remember, it's not possible to affect the nature of a piano note (other than to curtail it) once it has sounded. Any parameters that let you change the brightness, the loudness, or add vibrato by bearing down on a depressed key must be set to zero.

We now come to the two sounds comprising the Performance, and parameters 31 and 41 allow us to insert Piano 1B and Piano 1A into their appropriate slots. Next, parameters 32 and 42 shift the two patches down an octave (this may be a modification of my own, not Roland's original programming... I forget), while numbers 33 and 43 tell the JX10 that they respond in 'Poly 1' mode, which means that a new voice is assigned each time you press a key. (This also makes parameters 34 and 44 irrelevant, because you cannot be in a polyphonic mode and a Unison mode

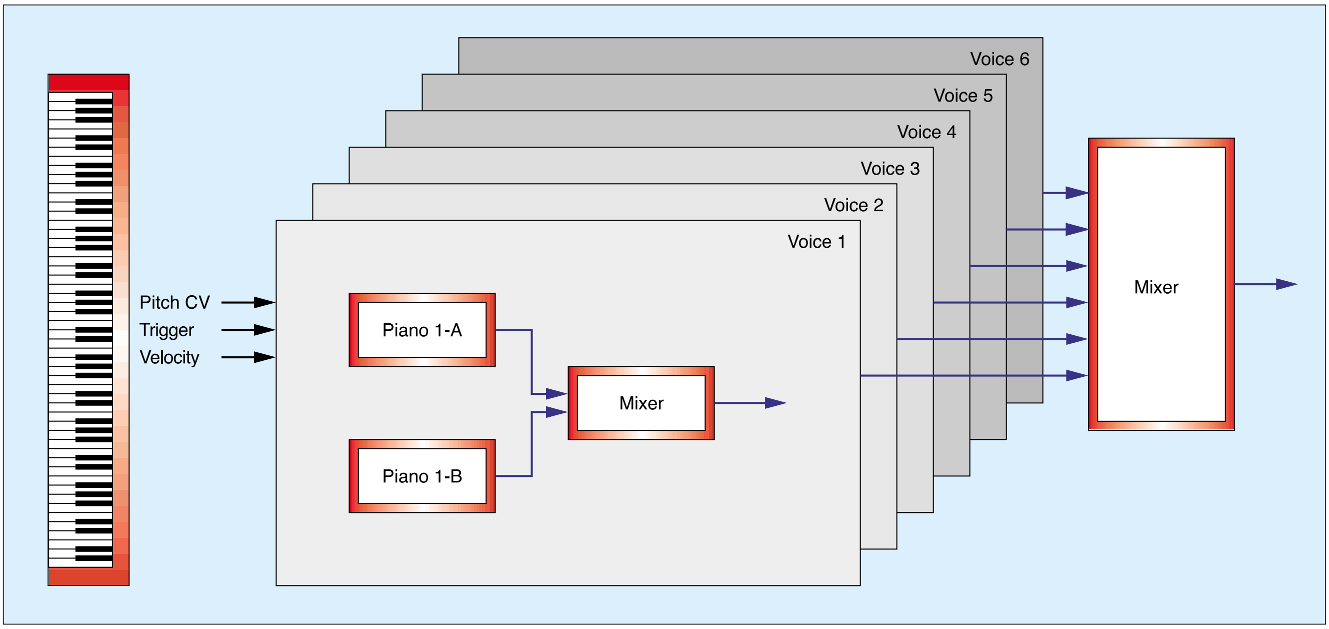


Figure 8: Layering two patches in 'Dual' mode.

simultaneously.) Next, we find that 'Hold' — the response to the sustain pedal — is On for both patches, Portamento (as stated above) is Off, and Bend (I still don't understand this) is On. That leaves LFO Mod Depth, which is the Performance's response to the modulation direction of the combined pitch-bend/modulation joystick. Again, this should be zero for both patches, but for Piano 1B it is set to 01. In truth, I find this imperceptible, but it should be zero nonetheless.

## The End Result

So what does this tell us? Stripping away all the superfluous bits and pieces, we have simply taken two similar, but not identical, six-voice patches and layered them at the same loudness across the keyboard, but with a small tuning offset. (See Figure 8.)

There's nothing particularly clever happening here; you could do the same thing by taking a MIDI synthesizer and connecting it to an equivalent module, playing the two simultaneously and mixing their outputs into a single sound. Given this, it's time to ask once again, "how does it sound?". The answer may surprise you. 'H1: Acoustic Piano' sounds more rich, more vibrant, more expressive, more like a real instrument. But why?

The secret — and it's an important one — lies in the combination of two sounds that are similar enough to be indistinguishable within the composite, but different enough to create a sound that is more interesting than either of the components in isolation. Look at it like this: if you layered and detuned the piccolo and Minimoog bass that I mentioned near the start of this article, the composite would sound like an out-of-tune piccolo and Minimoog bass. On the other hand, if you layered two detuned but otherwise identical sounds, the result would sound like the original, but chorused.

On the other hand (which I realise is only possible if you have three hands) the two components in 'H1: Acoustic Piano' complement each other in superb fashion. Piano 1B supplies the initial thunk, while Piano 1A has the richer spectrum and provides more of the body of the sound. Furthermore, the detuned harmonics of the complex, sync'd waveforms sweep in and out of phase with one another, reinforcing and then interfering with one another destructively, to imitate the energy interactions within an acoustic piano. Then, towards the end of the note, Piano 1B dominates again (thanks to the longer Decay and Release in ENV2, which drives the Gain of the audio VCA) and the filter closes to leave just the fundamental and a few low harmonics in the tail.

All of this conforms closely to the principles we derived for the piano in the October instalment of Synth Secrets. What's more, if you consider things such as the filter scaling and dynamics responses of the component patches, you'll see that Roland's programmers were not blindly groping for their piano sound: this performance was crafted with a great deal of thought.

So I'll ask one final time, "How does it sound?" The answer is that 'H1: Acoustic Piano' has many of the characteristics of an acoustic or electro-mechanical piano, without sounding anything like the former, or even quite like the latter. It's responsive, it's expressive and, for many purposes, it's every bit as usable as a Fender Rhodes 73 or a Wurlitzer EP200. In fact, there are times when I would still use it today, in preference to any of the 'real' things.

# Synth Secrets

## Synthesizing Strings • String Machines

Original photos: Richard Ecclestone

**Analogue synths can't synthesize every sound, but the attempts made to replicate the sound of orchestral strings were so successful that so-called string machines became much-loved instruments in their own right. We begin a voyage into the world of synthesized strings...**

Gordon Reid

For the past few months, I've been investigating the sound of the acoustic piano, and showing that when we attempt to synthesize this, the best we can manage is something that sounds similar to an electro-mechanical piano. Of course, this is a valid aim in itself, and I can think of numerous specialist synths dedicated to this task. Reviewing one of these recently set me thinking... are there other important electronic or electro-mechanical keyboard sounds that are themselves emulated using synths? Of course there are; almost every polysynth has patches designed to emulate Hammond organs and Hohner Clavinets. But just as important as these is a class of electronic keyboards that has provided the foundation for much electronic music from the 1960s onwards. There are the ensemble keyboards, known variously as string ensembles, string synths or simply string machines.

The first keyboard to fit the modern description of the Ensemble was the prototype Freeman String Synth. This used the divide-down oscillator technology of cheap 1960s organs, but sounded nothing like a cheap organ (for more on organs and divide-down technology, see the instalment of this series in *SOS* December 2000, or direct your web browser at: www.sound-on-sound.com/sos/dec00/articles/synthsec.asp). Indeed, its inventor, Ken Freeman, was sought-after as a session player in the 1970s because he was capable of creating superb ensemble arrangements using this keyboard.

Freeman had a distinct advantage over subsequent players because his instrument used more oscillators and more modulators per note than were employed in the cut-down version sold to the public. The Lowrey String Symphoniser (no, that's not a spelling mistake) was still a nice instrument, but Freeman's original produced a remarkably lush sound without resorting to chorus effects.

In theory, Freeman could have used multiple oscillators per note to imitate the sound of multiple musicians (such as violinists) playing at the same pitch. In addition, he might have tuned each oscillator differently to imitate the random inaccuracies of human players. He might even have added multiple envelopes and modulators to imitate the different articulation and vibrato of each player. And, in theory, that would have worked nicely. But given the size and weight of electronics from that era, and the fact that a human ensemble can comprise 30 or more musicians, the Freeman String Synth would have been huge, heavy, and ridiculously expensive. Even with today's digital technology, there has never been a polysynth that offers 30 oscillators, 30 envelopes, and 30 or more LFOs per note.

Clearly, Freeman's keyboard worked in a different way, so is it reasonable to assume that we can use any analogue polysynth to emulate its architecture and recreate its sound? Well... no. The Freeman string synth was fully polyphonic across its 61-note keyboard, and (if I remember correctly) incorporated no fewer than three oscillators and three modulators per note. This is non-trivial stuff, equalled (well, almost equalled) only by the mighty Korg PS3300. But since the Korg now commands prices approaching £10,000, I thought that it might be interesting to see whether we can use an affordable analogue polysynth to create a lush ensemble sound and — like Freeman himself — do so without recourse to effects units.

### Detuning Multiple Oscillators

Why is it a good idea to have a synth with more than one oscillator per voice? Well, it means you can mix two waveforms in a single sound (although many synths allow you to mix the waveforms produced by each oscillator), and you can also program intervals between oscillators. Many important sounds were invented this way, such as Keith Emerson's lead sounds (these were unimaginable before the advent of the Moog modulars in the 1960s, and used three oscillators tuned to the tonic, third and fifth).

But for me, the most frequent benefit of having two or more oscillators per voice is to be able to detune them. Programmed correctly, this does not give rise to two distinguishable pitches nor, if used judiciously, does it make one sound 'flat' against the other. Instead, the interaction of the two creates a new class of timbres.

Figure 1 (on next page) shows what happens when you mix two sine waves of equal amplitude, one of frequency 100Hz,

Figure 1: (left) The effect of summing two sine waves of similar frequencies.

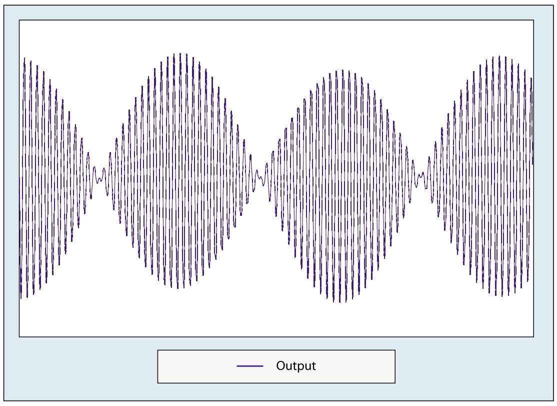
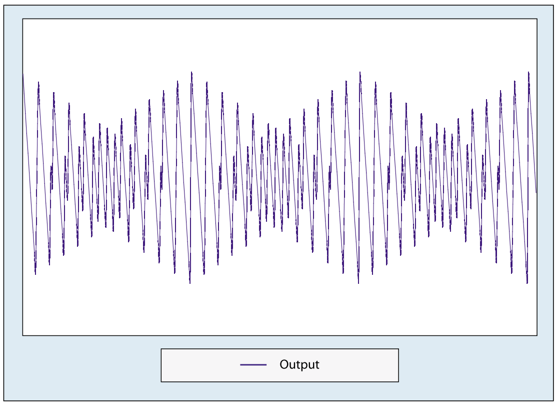


Figure 2: (Right) One possible waveform generated by detuning two sawtooth waves.



and the other of 101Hz. The result may look surprising, but it is obvious when you think about it. The two signals are moving in and out of phase with each other so there will be times when they reinforce and there will be times when they interfere destructively. Consequently, the amplitude of the resulting waveform will range from the sum of the inputs to zero, swinging smoothly between the two.

Looking at Figure 1 again, it's clear that the output is punctuated by silences. We hear this as *beating*, and its frequency is the same as the difference in the frequencies of the two oscillators.

Beating is familiar to everyone who has ever played with the detune knob on a dual-oscillator synth, but it may not be obvious how this leads us toward ensemble sounds. So it's time to remind ourselves that audio-frequency sine waves are quite rare, and that the waveform produced by a string ensemble — whether human or transistorised — is going to be significantly different from this.

The unmodified waveform produced by a real string is similar to a sawtooth wave, so it's no surprise to find that the waveform selected by the designers of electronic ensembles was always a sawtooth or a similarly spiky waveform. When we calculate the result of detuning two sawtooth waves, we find that these exhibit a less pronounced modulation, one form of which I have shown in Figure 2. Listening to this, we find that the new sound appears 'thicker' than either of the oscillators in isolation, and this can be very pleasing to the ear.

## Synthesizing The Ensemble Sound

The sound in Figure 2 is still nothing like that of a string synth, although it's much more interesting than that of a single oscillator in isolation. So let's now ask two questions: Can we affect the thickened sound in Figure 2 to create something that sounds like an ensemble keyboard? And can we do this on an affordable analogue polysynth?

The answer is 'yes' in both cases, and, to illustrate this, I'm going to use a dual-oscillator polysynth that I happen to have by my left hand: the Roland Jupiter 6.

I'll start by selecting the sawtooth waveform for VCO1, and I'll give it a pitch of 8'. If I turn the Mixer fully anticlockwise so that I hear just VCO1, and set the filter wide open with no enveloping and no modulation of any sort, the result proves to be a boring buzz, much like the cheapest, nastiest transistor organ imaginable (see Figure 3). So now I'm going to add the second, detuned oscillator to the patch, as shown in Figure 4.

Setting VCO2 to be another sawtooth wave and detuning it with respect to VCO1, I find that — contrary to what I wrote a few

▶

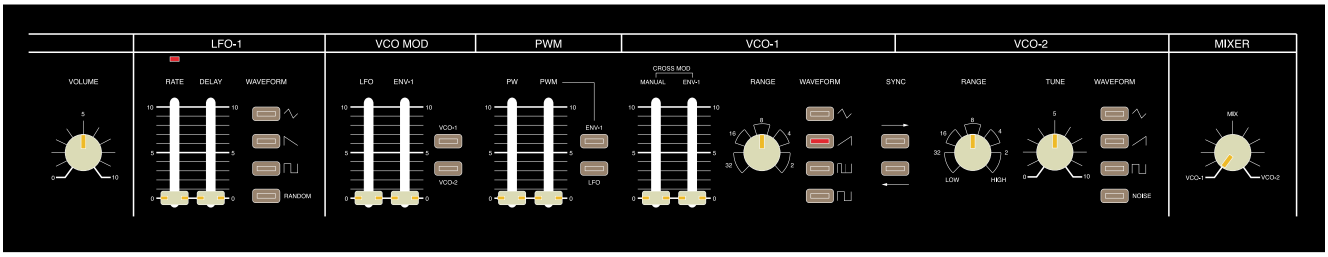


Figure 3: A single sawtooth produces a boring buzz.

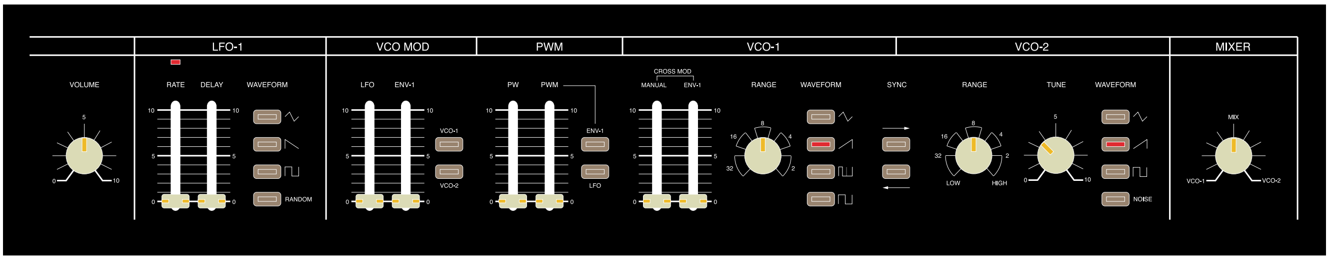


Figure 4: Adding a second, detuned oscillator to the sound.

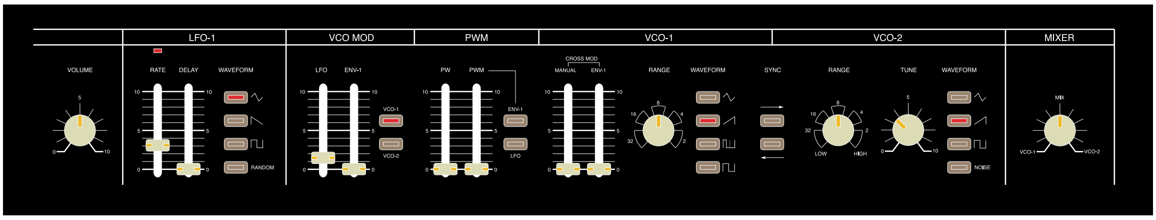


Figure 5: Adding pitch modulation to just one oscillator.

moments ago — the result is still rather boring. Sure, there's a degree of movement imparted by the detune, but this isn't particularly impressive. Increasing the amount of detune increases the movement, but pretty soon the timbre starts to sound strange, with a weird 'off colour' timbre. This might be suitable for programming honky-tonk pianos, but it's quite inappropriate for emulating string ensembles.

Bearing this in mind, I'll choose a minimal amount of detune, and use the triangle wave produced by LFO1 to add a little vibrato to VCO1 (see Figure 5, above). Does this improve matters? Yes, it does, especially when the beat frequency of the detune is different from the LFO speed. Whereas VCO1 in isolation has been transformed from a boring buzz into a wobbly boring buzz, and VCO2 remains a boring buzz, the mixed sound — which, to be honest, is still horribly buzzy — has assumed a richer tone than before. So far, so good...

## Multiple Sine Waves & Modulation

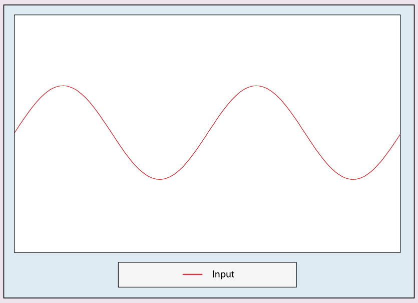
**As we know, ensemble sounds require a great deal of modulation... far more than the simple vibratos and tremolos generated by a single LFO. Ideally, we would like to modulate each oscillator in an ensemble patch independently so that the differences in their frequencies are varying in a convoluted fashion. Unfortunately, this is rarely possible on affordable synths of any kind. But you might wonder whether it's possible to leave one oscillator unaffected and modulate the other in a more complex way that will fool the ear into thinking that there are multiple modulations present. After all, surely you don't need banks of LFOs to create complex modulation waveforms? It seems reasonable to speculate that we could take a single sine wave, direct it down a number of signal paths, delay some with respect to others (ie. change their phases) and mess with their amplitude before mixing the various elements to generate complex waveforms.**

**Sadly, this is not the case. I can demonstrate this by taking the simple sine wave in the first diagram below, and adding another instance of the same waveform as shown in the next diagram. As you would expect, the result is another sine wave, but with double the amplitude.**

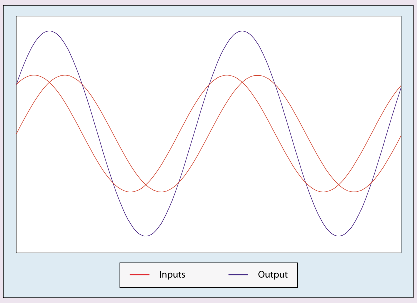
**I'll now offset one of the waves in this second diagram by shifting its phase relative to the other. Intuitively, we might now expect a 'wiggly' waveform, but the third diagram (see or below) shows that the output is another sine wave. Well, how about giving one wave a greater amplitude than the other, as shown in the fourth diagram No... the output remains a sine wave. If fact, we can continue to add as many sine waves as we wish, and with whatever phases and amplitudes as we wish, but if their frequencies are the same, the result is always another sine wave. If we want to generate more interesting waveforms we must mix two sine waves of different frequencies, as shown in the fifth diagram.**

**We can make the modulating waveform in the fifth diagram even more complex if we add a third (or fourth, or fifth...) sine wave, as shown in the last diagram in this box. The output then loses all semblance of periodicity and, depending upon the ratios of the frequencies of the components, it may not repeat for the whole duration of a given piece of music. This means that the modulator is — as far as we can hear — aperiodic, which can be useful when we program sounds with a quasi-random or 'human' element within them.**

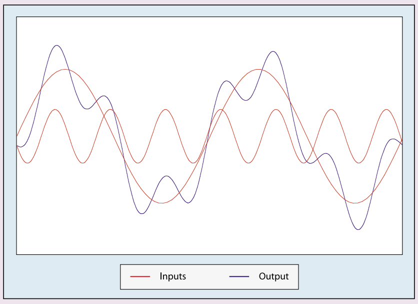
**Sadly (again), it's rare to encounter an analogue polysynth with such modulation capabilities. Roland's Jupiter 6, however, is one of very few analogue polysynths that has this ability. In addition to LFO1, you can apply the performance modulator, LFO2, to the VCO1 pitch, and then set VCO2 to 'Low', and apply it as a third LFO using the Manual Cross Mod slider in the VCO1 control panel. However, we will not make use of this in this month's patch because, in using VCO2 as an LFO, we lose the ability to create the detuned sound that is vital to the ensemble sound.**



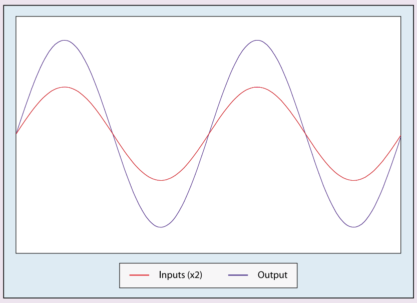
A simple sine-wave LFO.



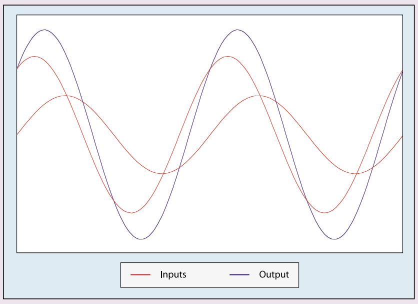
Adding out-of-phase sine waves creates... another sine wave.



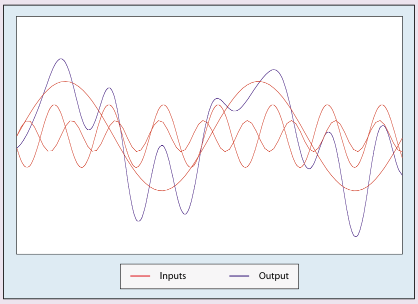
Adding two sine waves of different frequencies generates more complex waveforms.



Adding identical sine waves creates... another sine wave.



Adding out-of-phase sine waves of different amplitudes creates... another sine wave.



Adding three sine waves of different frequencies further complicates the output.

## Humanising The Sound

The reason for the improvement in the tone is simple to describe, although it would be a nightmare to illustrate. In brief, the LFO is altering the amount of detune between VCO1 and VCO2, thus changing the rate of the modulation shown in Figure 2. It is this that our ears hear as the further thickening of the sound. However, when you listen to the patch more carefully, you will hear an unnaturally regular modulation within it. This is caused by the constant nature of the triangular output from the LFO and, while it is not unpleasant, it would be interesting to see whether we can eliminate it without damaging the detuned effect itself.

When the synth in question has a single LFO limited to simple waveforms, there's nothing we can do. However, the Jupiter 6 offers an LFO setting called Random. This is a sample & hold generator whose clock rate is the LFO rate, and whose sample source is an invisible noise generator (see Figure 6, below).

Let me now direct you back to *SOS* August 2000, and the instalment of this series therein. Alternatively, head for www.sound-on-sound.com/sos/aug00/articles/synthsec.htm. This article illustrated how the sample & hold circuit extracts samples from a varying waveform. If the source is a noise generator, the output looks much like the blue waveform in Figure 7 (above)... and it is this that Random offers you.

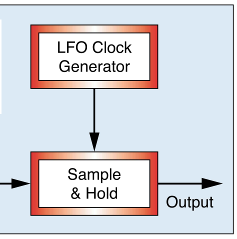


Figure 6: A simple representation of a 'random' circuit.

Now, I'd like to refer you to the box on modulation on the previous page, and in particular to the last diagram in the box, which shows a waveform similar to the output in Figure 7. Sure, the Random output is 'square' rather than smooth, but the essential character is the same. This implies that we can use Random to modulate VCO1 and remove the predictable nature of the interactions between the oscillators.

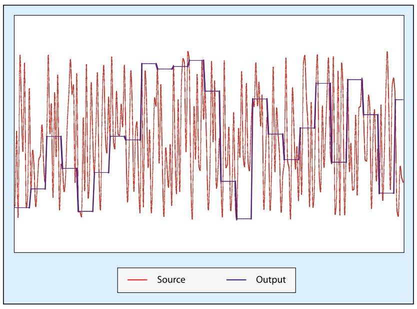


Figure 7: The output from a S&H module with noise as the source waveform.

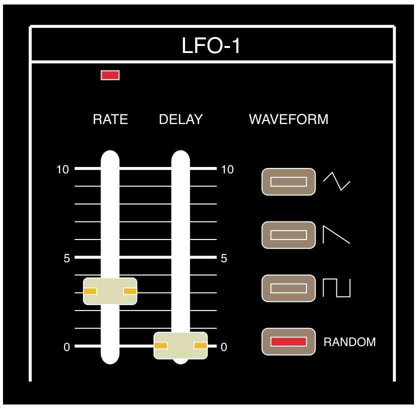


Figure 8: Selecting a random wave for the VCO1 modulator.

If you make this change, as shown in Figure 8 (above) you will hear that the essential nature of the sound remains the same, but that there is now no periodic modulation. If you increase the LFO Depth too far, you will hear the pitch of VCO1 jump around in a most unnatural fashion, but with minimal modulation, the 'square' nature of the Random setting is masked and we hear something that sounds thick and unstable... 'analogue', or perhaps 'human', whichever way you look at it.

## Shaping The Sound

Having programmed the above, we can now proceed to shape the sound using the filter, amplifier and envelopes. Given that you may not be following this exercise on a Jupiter 6, I can't give you precise settings for the other controls, but I can point you in the right direction with the following pointers:

- If your polysynth is velocity-sensitive, all "Dynamics' parameters should be set to 'Off'. String synths were not velocity-sensitive, so this patch should be likewise.
- In general, the low-pass filter should be about halfway closed, with a moderate amount of key-follow (ie. so that the sound becomes somewhat brighter as you move up the keyboard). There should be no filter modulation.
- The VCA Gain should be shaped by a trapezoid contour that generates the crescendo, as well as the extended tail of the sound (see Figure 9 below).

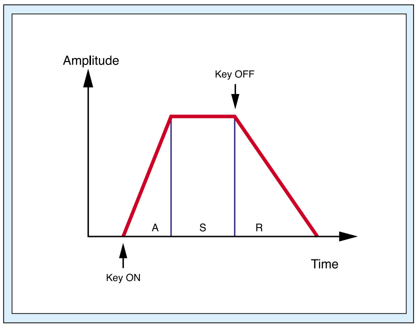


Figure 9: The trapezoid envelope shape for the VCA.

Figure 10 (below) shows the VCF and VCA sections of the Jupiter 6 set up appropriately.

Your patch should now sound like a typical 'pad', the type that I often call a 'carpet' because it is used to provide a thick, fuzzy foundation for many songs. Nonetheless, the sound still lacks the depth and character of a true string synth.

With many polysynths, this will be as close as you can get to the ensemble sound, and, if you're using one of these and still require an extra 'sheen', your only recourse will now be to use an external chorus unit. This does not need to be expensive, and my favourite mono in/stereo out chorus pedal, the Roland Dimension C, turns variations of this patch into all manner of classy string sounds, such as the ARP Omni strings from Pink Floyd's 'Shine On You Crazy Diamond'.

But to me, this feels like 'cheating' of a kind. I still want to see whether I can create Freeman's string ensemble sound without using effects, as he did, so I'm now going to invoke another trick that the Jupiter 6 has tucked up its electronic sleeves...

## Pulse Width Modulation

I mentioned in passing near the start of this

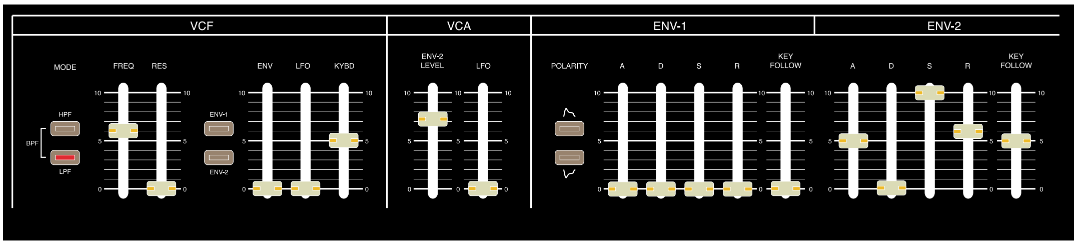


Figure 10: The VCF and VCA sections for our string-synth patch.

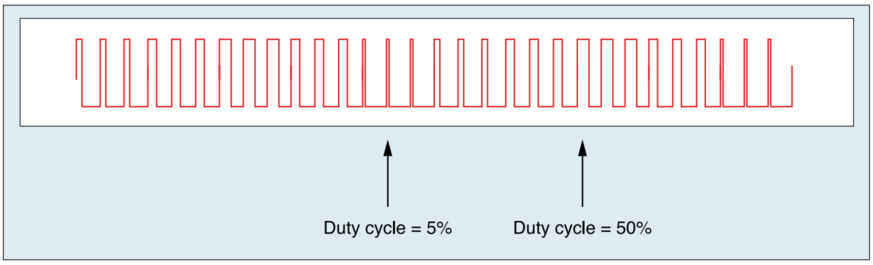


Figure 11: A pulse-width-modulated pulse wave.

article that many synths allow you to mix the waveforms produced by each oscillator, and the Jupiter 6 is one of these. This means that we can mix the sawtooth waves and Pulse Width Modulated (PWM) pulse waves produced by each of VCO1 and VCO2.

Figure 11 (above) shows the PWM waveform generated by using a triangle wave to modulate the duty cycle of a pulse wave between 50 percent (a square wave) and about five percent. You may recall from earlier instalments of this series (see *SOS* February 2000, or go to: www.sound-on-sound.com/sos/feb00/articles/synthsecrets.htm) that at any given moment, the harmonic content of a pulse wave is dependent upon this 'duty cycle': ie. the ratio of the length of the flat bit at the top of the wave to the length of the bottom bit. Consequently, the harmonic content of the PWM wave is continuously changing, and many writers have stated that this is why PWM waves have a more interesting sound than those produced by static waveforms.

To be fair, the changing harmonic content is one of the visible consequences of the underlying reason behind the lush sound of PWM waves, but there's more to it than that. This is a complex topic in its own right, so I will return to it next month. But for now, take my word for the following.

Pulse waves whose widths are modulated by triangle waves have another, rarely appreciated characteristic; they exhibit pitch modulation that oscillates at the PWM rate above and below the true oscillator pitch. If you have a suitable synth to hand, it's easy to hear this at low oscillator pitches and high modulation depths but, if not, I'll ask you to take my word for it that a PWM wave generated by a single oscillator exhibits *two* pitches, as shown in Figure 12. It is easy to see that these detuned pitches will give the PWM wave its characteristic 'chorused' sound.

Since the Jupiter 6 has two oscillators with PWM settings, we can detune and mix these to create the sound represented in Figure 13. There are now four pitches in the output, so it's easy to believe that the result will be far richer than anything we have created so far (if we could modulate the oscillators with independent rates and depths, that would be even better... but let's not get carried away).

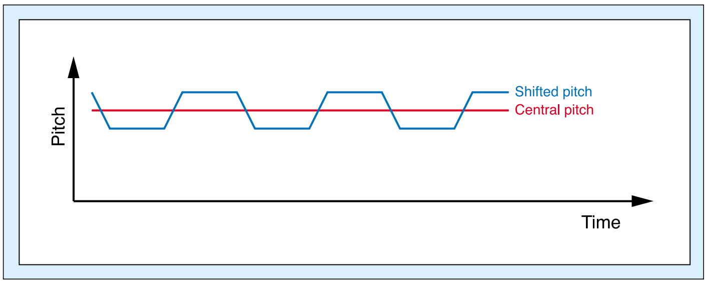


Figure 12: A PWM wave generates two pitches simultaneously.

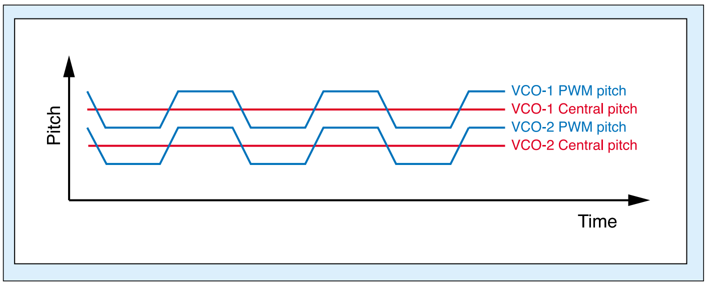


Figure 13: Mixing two detuned PWM waves to generate four pitches simultaneously.

To create this sound, we return LFO1 to a triangle wave, increase the rate slightly, and then apply the pulse-width modulation to the oscillators using the PWM slider and LFO button in the PWM section. Listening to the result, I find that it has a slightly 'hollow' flavour, so we can now invoke the Jupiter 6's ability to mix both pairs of sawtooth and PWM waves by pressing the sawtooth and PWM buttons simultaneously on VCO1 and VCO2 (see Figure 14).

Direct the output from this through the VCF and VCA settings in Figure 10, and... Bingo! It's thick, it's lush, and it's certainly an ensemble sound. Of course, it sounds nothing like real orchestral strings, but that's not the point. As most keyboard players have discovered over the past three-and-a-bit decades, string synths have a characteristic sound all their own, and it's one that has musical uses just as valid as any other instrument.

Before saving or leaving this patch, there's nothing stopping you from passing it through a chorus unit to make it even richer. You might also like to lower VCO2 by an octave because this adds 'weight' without changing the fundamental character of the sound. Alternatively, you could increase the Attack and Release settings in ENV2 for slow, dreamy pads. Or you could... Oh heck, there are thousands of variations. Discover them for yourself!

Now, I'll admit that, in using pulse-width modulation, I've strayed beyond Ken Freeman's original ideas, but I have no remorse... There is no alternative if you want to get the Jupiter 6 to jump through this particular hoop. Interestingly, the Roland Super JX10 (which I used to help illustrate my piano sounds over the past few months) is capable of a far better ensemble sound than the Jupiter 6, and its architecture doesn't include pulse-width modulation. I would love to show you how, but I've already used up all my space this month, so I'll have to leave that to another day. SOS

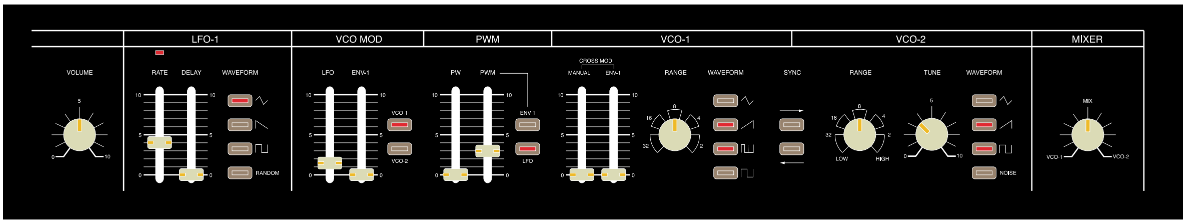


Figure 14: Using pulse-width modulation to create an even richer ensemble sound.

# Synth Secrets

## Synthesizing Strings • PWM & String Sounds

Original Photos: Richard Ecclestone

**Pulse-width modulation is a vital tool in achieving lush-sounding synthesized string pads — so what if your synth doesn't have it? Fear not — for PWM can *itself* be synthesized. Here's how...**

Gordon Reid

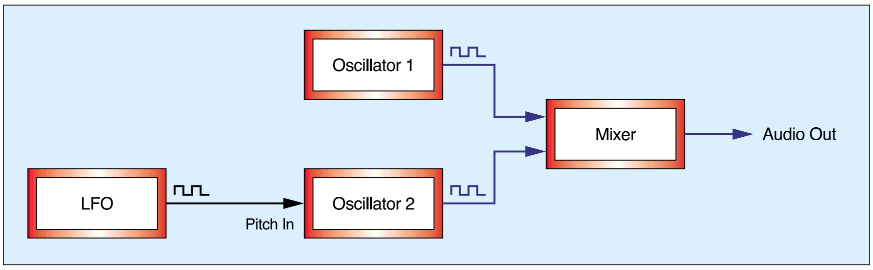
Last month, we started investigating how to synthesize strings, and the wonderful lush string sounds produced by the dedicated analogue ensemble keyboards of the 1970s. In doing so, we found that the chorusing of two slightly detuned and/or modulated oscillators was an important part of their sound; one that we could emulate using pulse-width modulation (or PWM).

During this investigation, I made a couple of statements about the nature of PWM and the audible effect it has on a previously static pulse waveform. Without any kind of proof, I asked you to accept that pulse waves whose widths are modulated by triangle waves have an interesting and largely unknown characteristic; they exhibit *two* pitches, one of which undergoes pitch modulation that oscillates at the PWM rate above and below the true oscillator pitch. I said I'd return to this point this month, and I will, because it will allow us to come full circle and produce some amazing ensemble sounds on synths that you might think incapable of generating them.

It is, in fact, relatively simple to create a close approximation of a pulse-width-modulated sound, even on a synth that doesn't offer PWM. By way of proof, this month I'll take one of the most 'digital-sounding' of all the synth engines derided by vintage-synth anoraks — the 'AI' synthesis used in Korg's famous M1 workstation — and create a sound that is truly 'analogue' in warmth and character.

If you wish to know some of the theory behind the method that makes this possible, take a look at the box on the nature of the pulse waveform on pages 154-155, and then at the larger box on analysing PWM on pages 156-157. If you're not bothered about the theory, you'll simply have to accept that when they're combined in the right way, the 'staircase' waveforms shown in Figure Z at the end of the large box form the perfect pulse-width-modulated pulse wave. It's from this point that we'll now begin.

Figure 1: Trying to recreate PWM using two square waves.



### PWM From Combined Waveforms

As explained in the box on page 156-157, the two staircase waves that comprise PWM extend to infinity, so it's clear that no *analogue* synth can generate them. This is because they would require internal voltages whizzing off towards infinity and minus infinity (and I don't know how *you* feel about having infinite voltages floating around inside your synth, but I would rather avoid it). The situation is no better if we turn to digital synthesis. Instead of infinite voltages, we need infinite wordlengths. This might lead you to believe that it's impossible to generate the PWM waveform without access to PWM itself, but fortunately, this is not the case.

Given the nature of PWM and the 'staircase' waves shown on page 157, it seems reasonable to start any attempt to synthesize PWM using a couple of square-wave oscillators, so we'll adopt the architecture shown in Figure 1 (above) to modulate and combine two square waves in the way that we know characterises pulse-width modulation. Unfortunately, we obtain the waveform shown in Figure 2 (below). It's an interesting wave, with a sort of 'double PWM' above and below its central amplitude. What's more, it exhibits a pleasing chorused quality, but it's audibly *not* PWM, and I'd like us to get closer to the ideal. If you have a basic synth and an oscilloscope, you could try combining sawtooth waves,

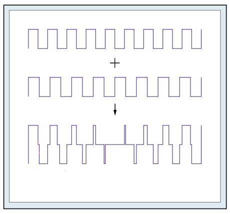


Figure 2: An idealised representation of the signal generated by the architecture in Figure 1.

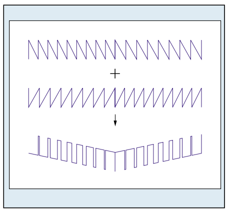


Figure 3: Emulating PWM with modulated sawtooth and ramp waves.

or even square and sawtooth waves, but the results are no better. However, if you have one of the few instruments that offer sawtooth and ramp waves, you could try combining these. I used analogue modules present in my Analogue Systems Sorceror modular synth — two RS90 oscillators (which offer sawtooth and ramp waveforms), an RS380 modulation controller, and an RS160 mixer, to be precise. Bingo! When patched and programmed correctly, the result looks just like PWM, as shown in Figure 3 (below left).

Note that Figure 3 shows a low-frequency amplitude modulation at the LFO speed (hence the downward and upward slope to the resulting PWM wave). Don't worry about this; the prominence of the modulation in the diagram is a consequence of the limited space available for the drawing. If the audio waves are pitched at a few hundred Hertz (somewhere in the middle of the piano keyboard) and modulated by an LFO running at a fraction of a Hertz, this modulation is of no significance, provided that you allow sufficient headroom to accommodate it. If you want, you can use a high-pass filter to remove it, but it's rarely worth the effort.

## Creating PWM... Without PWM

Although it's not strictly kosher to do so, I am now going to take advantage of a trick that will allow us to emulate the PWM sound on synths less complex than the Sorceror. In isolation, the audible difference between a sawtooth wave and a ramp wave is, well... inaudible. This is because they are static waves that have the same spectrum; there is merely a change of polarity. There are times when this difference can be significant, but it is of little consequence at others. Luckily, the two oscillators generating the waves in Figure 3 are never of the same frequency, so this is one of those occasions when you can use a sawtooth to replace a ramp wave with no audible effect. This, of course, is a good thing, as many more synths offer sawtooth waves than ramp waves. In other words, you can mix two simple sawtooth oscillators, and, if one is frequency-modulated slightly with respect to the other, you will obtain a sound that is all but identical to that of a single, pulse-width modulated, pulse-wave oscillator. Sure, the waveform looks different, but the sound is the same.

Knowing this, we can now create PWM 'string' pads on a synth that has no PWM. As promised, to illustrate this, I have taken the Korg AI engine, as found in Korg's M1 and 'T'-series, and programmed something that is warm and, dare I say it, quite 'analogue' in nature. Given that the M1 is often criticised for being 'soulless and digital' (whatever that means) this was deliberate; after all, if we can use the analyses in the boxes in this article to achieve a convincing imitation of a string ensemble using AI, the theoretical principles must have real practical value.

To begin, I took my Korg T2, which is an M1 with a longer keyboard and a decent screen, and programmed the oscillator parameters as shown in Table 1 below.

**KORG T2 OSCILLATOR PARAMETERS**

| | |
|---|---|
| OSC Mode | Double |
| Assign | Poly |
| Waveform 1 | Sawtooth |
| Octave | 16' |
| Waveform 2 | Sawtooth |
| Octave | 16' |
| Interval | 0 |
| Detune | 0 |
| Delay | 0 |

Table 1.

I then moved on to the twin filters (one for each oscillator) and set the cutoff of each to '75', with a little keyboard tracking. All other filter parameters remained at zero. These settings emulate the limited bandwidth and simple filtering on the original string synths.

Moving on to the T2's twin amplifiers, I defined each with a trapezoid envelope exhibiting a rapid but smooth attack, and a gentle release (see Table 2 below).

**KORG T2 AMPLIFIER/CONTOUR PARAMETERS**

| | |
|---|---|
| Velocity Sense | 00 |
| KBD Tracking | -04 |
| Attack Rate | 10 |
| Attack Level | 99 |
| Delay Time | 99 |
| Break Point | 99 |
| Slope Time | 99 |
| Sustain Level | 99 |
| Release Time | 49 |

Table 2.

The only subtle parameter I used was keyboard tracking. With a negative value, this weights the loudness of the sound to the bottom end of the keyboard, thus generating additional warmth. I set all the other parameters on this page to zero.

The next page contains the Modulation parameters and, ignoring things such as pitch-bend and aftertouch, it was here that I defined the pitch modulation of Oscillator 2. The relevant parameters are in Table 3 below.

**KORG T2 MODULATION PARAMETERS**

| | |
|---|---|
| Pitch MG Waveform | Square |
| Frequency | 60 |
| Intensity | 02 |
| Target | OSC2 |
| Sync | OFF |
| Aftertouch | OFF |
| Joystick Intensity | 00 |
| Joystick Frequency | 00 |

Table 3.

**SUPER JX10 PARAMETERS**

| | | |
|---|---|---|
| **DCO1** | | |
| 11 | Range | 8' |
| 12 | Waveform | Sawtooth |
| 13 | Tune | 00 |
| 14 | LFO Depth | 02 |
| 15 | Envelope Depth | 00 |
| **DCO2** | | |
| 21 | Range | 8' |
| 22 | Waveform | Sawtooth |
| 23 | Cross Modulation | OFF |
| 24 | Tune | 00 |
| 25 | Fine Tune | -04 |
| 26 | LFO Depth | 00 |
| 27 | Envelope Depth | 00 |
| **LFO** | | |
| 71 | Waveform | Square |
| 72 | Delay | 00 |
| 73 | Rate | 78 |
| **MIXER** | | |
| 41 | DCO1 | 99 |
| 42 | DCO2 | 99 |
| 43 | Envelope Depth | 00 |
| 44 | Dynamics | OFF |
| 45 | Envelope Mode | — |

Table 4.

**SUPER JX10 FILTER, VCA, & ENVELOPE PARAMETERS**

| | | |
|---|---|---|
| **VCF** | | |
| 51 | HPF | 0 |
| 52 | LPF Frequency | 47 |
| 53 | LPF Resonance | 00 |
| 54 | LFO Amount | 00 |
| 55 | ENV Amount | 14 |
| 56 | Key Follow | 64 |
| 57 | Dynamics | OFF |
| 58 | Envelope Mode | ^1 |
| **VCA/CHORUS** | | |
| 61 | Level | 67 |
| 62 | Env Mode | E2 |
| 63 | Dynamics | OFF |
| 64 | Chorus | OFF |
| **ENV1** | | |
| 81 | A | 15 |
| 82 | D | 00 |
| 83 | S | 99 |
| 84 | R | 39 |
| 85 | Key Follow | OFF |
| **ENV2** | | |
| 91 | A | 45 |
| 92 | D | 00 |
| 93 | S | 99 |
| 94 | R | 54 |
| 95 | Key Follow | |

Table 5.

And that's all there is to it. If you have access to an M1 or a 'T' -series synth, you'll find that it's definitely a PWM-type sound, despite being created on an instrument that does not offer PWM. Sure, the result still lacks a little of the warmth of, say, the PWM found on a Roland Juno 60, but not to worry... we can improve the patch considerably by detuning Oscillator 2 by a value of '10', or thereabouts (the eighth

▶

parameter in Table 1). This is something that you cannot do on a Juno!

Of course, there's nothing stopping you using the Korg's onboard effects to thicken the sound still further. Careful use of EQ allows you to shape the sound, and the addition of chorus and a splash of reverb leaves me in no doubt that I could fool you into thinking that my T2 is a vintage ensemble keyboard, and not a digital workstation. This should not be surprising; most string synths relied heavily on built-in chorus effects to thicken a weedy initial timbre. In contrast, what we're doing here is adding chorus to a basic tone that already has many of the characteristics we want, and sounds superb.

## More On Strings & Layering

A few months ago, I showed that you can create excellent sounds by layering patches to create a more interesting composite. The T2 is capable of this, and it's simple to create two similar versions of the PWM patch, mixing them and passing them through the effects to create an even deeper and more involving sound. You could even use three patches, tuning these to 16', 8' and 4', and mixing them as desired to emulate the Cello, Viola and Violin options offered by some of the better vintage ensembles. Admittedly, the polyphony drops to eight voices (two layers) or five voices (three layers) but the resulting sounds are so lush that I doubt you would want more.

Fortunately, layering is not a facility limited to *digital* synths and workstations. Just as we recently superimposed 'Piano 1-A' and 'Piano 1-B' to create an acoustic piano patch on the Roland Super JX10 (see the instalment of this series in January's *SOS*), we can use the same principle to create some of the world's best string pads.

Selecting the JX10's 'Whole' mode, we'll start by programming the first Patch in a Performance, setting DCO1 and DCO2 to the values shown in Table 4 (on the previous page). As you can see, both oscillators are producing sawtooth waves, with DCO2 detuned by four cents compared

## The Nature Of The Pulse Waveform

**What is a Pulse wave? We all know what one looks like (you can see several in the larger box over the page if you need a reminder), but how can we break it down further? One way is to look at its harmonic content, as we did right back at the very start of this series, when we considered the various waveforms found on common analogue synths. To refresh your memories about how to describe static waveforms in this way, let's first look at the harmonic content of a sawtooth waveform.**

**If you've been following this series, you'll probably remember that the sawtooth wave has every harmonic present, with the amplitude of a given harmonic 'n' equal to 1/n times the amplitude of the fundamental. Figures A(i) and A(ii) below show this wave and its spectrum.**

**Note that rather than draw idealised representations, I have calculated all the waveforms and spectra shown here using 120 harmonics. If the fundamental were, say, 200Hz (a little below middle 'C') the highest harmonic would lie at 24kHz, which is well above the bandwidth of most synths (and your ears).**

**Now let's move on to the square wave. Although the shape of this is quite distinct from the sawtooth, its spectrum is similar — in fact, it appears to be the same, except that every even-numbered harmonic is missing. See Figures B(i) and B(ii) below.**

**As has been pointed out many times before, the square wave is simply a special case of the general family of pulse waves, distinguished only by the fact that the wave spends the same amount of time at the top of its cycle as it does at the bottom. We refer to this ratio as the 'duty cycle' of the pulse wave, and since the square wave spends 50 percent of its time at 'the top', we say that it has a duty cycle of 50 percent. Of course, there's nothing stopping us from creating pulse waves with other duty cycles, from 0 percent (where the wave spends *none* of its time 'at the top', and is therefore a silent DC signal at the minimum voltage) to 100 percent (where it is another silent DC signal, but this time at the peak voltage). Many synths allow you to program the pulse width in a range that lies typically between 5 and 95 percent. Others, such as the Minimoog, offer a number of fixed options. Either way, these produce a huge range of tones that are vital for imitating instruments as diverse as woodwind and bass guitars.**

**So let's now look at the waveform and harmonic content of other pulse waves, and in so doing correct a long-standing mistake usually made in discussions of pulse waveforms. It is often said that the harmonic spectrum of a pulse wave with a duty cycle of 33.3 percent is the same as that of the sawtooth wave, but with every third harmonic missing, and that the spectrum of a pulse wave with a duty cycle of 25 percent is the same as that of the sawtooth, but with every fourth harmonic missing... and so on. However, this is not the case, as we will now see.**

**Figure C(i) below shows a pulse wave with a duty cycle of 33.3 percent, and Figure C(ii)**

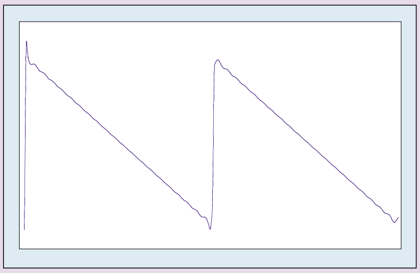


Figure A(i): The sawtooth wave.

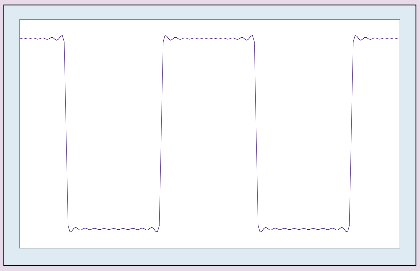


Figure B(i): A 'real' square wave.

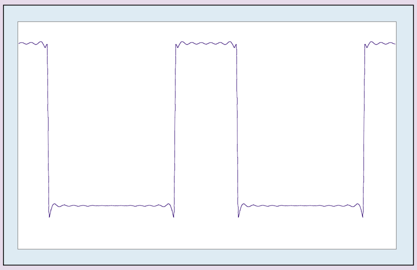


Figure C(i): A pulse wave with duty cycle of 33.3 percent.

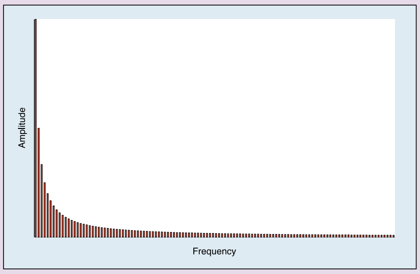


Figure A(ii): The harmonics that produced the wave in Figure A(i).

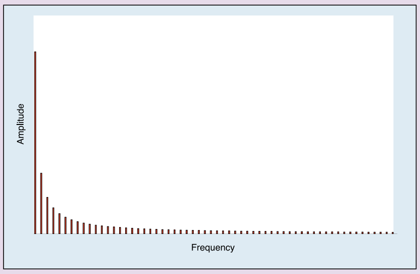


Figure B(ii): The harmonics that produced the wave in Figure B(i).

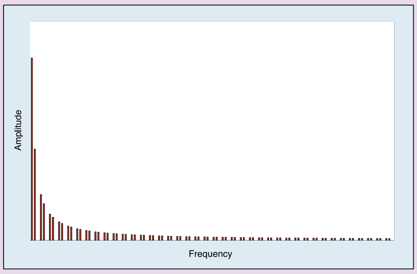


Figure C(ii): The harmonics that produced the wave in Figure C(i).

with DCO1. And, as on the Korg T2 example of a moment ago, we add a smidgen of 'square' pitch modulation to create the PWM effect.

The initial sound then passes through the filters and the VCA. The parameters in Table 5 (also on page 153) conform to the concepts described over the past couple of months, with all the JX10's 'modern' facilities switched off... this Patch requires only the facilities found on basic analogue synths.

So, how does it sound? Remarkably, given the low second-hand price of a JX10, this Patch is *much* nicer than the ensemble sound I created last month on the much more expensive Jupiter 6. In my opinion, it's smoother, stringier and much more useable in a mix.

But we need not stop there. Remember that although the Jupiter 6 offers just two oscillators per note, and Freeman's original String Synthesiser (which is where this discussion began) had three, we can now go further by taking advantage of the JX10's Dual Mode; its ability to layer two sounds in a Performance. I'll leave it to you to design the precise nature of the second Patch... but you should consider each of the following changes.

- Changing the LFO speed and depth alters the perceived PWM effect, and when combined with the first Patch, this enriches the sound by ensuring that the ear cannot detect any obvious periodic modulation.

- Altering the relative pitches of DCO1 and DCO2 changes the depth of the chorus effect.

- Adjusting the filter cutoff and envelope settings slightly will enrich the sound, because the ear will be aware that there is something more complex than a single sound at work, although it still won't be capable of separating the Patches from one another.

Having programmed the second Patch, you

**shows its spectrum. As you can see, this *does* appear to be the same as the spectrum of the sawtooth wave, but with every third harmonic missing... so where's the mistake?**

**Well... Figure D(i) below shows a pulse wave with a duty cycle of 25 percent, and Figure D(ii) shows its spectrum. *Now* you can see the difference. True, every fourth harmonic is missing, but the amplitudes of the others no longer exhibit the 1/n relationship. This becomes increasingly apparent as the duty cycle becomes narrower; if you're in any doubt, take a look at the spectrum for a pulse wave with a duty cycle of one-twelfth (8.33 percent) shown in Figure E (right).**

**The pulse wave's spectrum is defined by something called a 'sinc' function. This has nothing to do with synchronisation, or domestic water basins with plugs at the bottom, for that matter. Sinc is a mathematical function described by the simple equation shown in the top right of Figure F (below right). Without investigating this any further, it's enough to understand that the amplitude of any pulse-wave harmonic is defined by the value of the sinc function at that point (see Figure G below). Of course, you can't have negative amplitudes (or not for the purposes of this discussion, anyway), so we take the absolute value of each amplitude shown in Figure G, and thus we obtain Figure E again.**

**Looking more closely at these diagrams, it's clear that there can be no harmonics at the points where the sinc function crosses the 'X' axis, which explains why a pulse wave's missing harmonics are evenly spaced. It also explains why most people's attempts to create pulse waves using additive synthesis are doomed to failure. I have illustrated this by generating a 33.3 percent pulse wave as 'a sawtooth spectrum with holes in it'. As you can see, the spectrum in Figure H(ii) below may look no different from that in Figure C(ii), but its waveform, shown in Figure H(i) below, is a repeating 'staircase', with the number of steps equal to the spacing between missing harmonics. Remove every third harmonic (as illustrated) and you get three 'steps' in the staircase; remove every fourth, and you get four steps... and so on. If you removed every hundredth, accepted wisdom would suggest that you would obtain a narrow pulse wave (ie. with a duty cycle of one percent) but in reality, you obtain a staircase with one hundred small steps, which, if you think about it, is all but indistinguishable from a sawtooth wave. Given their nearly identical spectra, this is hardly surprising!**

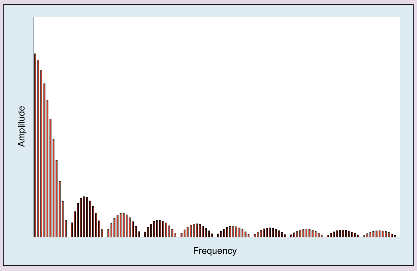


Figure E: The harmonics that produce a pulse wave with a duty cycle of 8.33 percent.

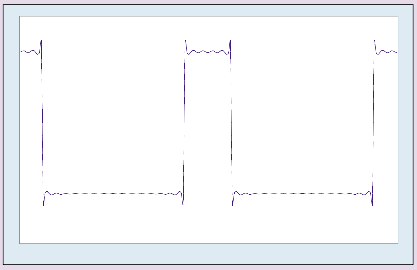


Figure D(i): A pulse wave with duty cycle of 25 percent.

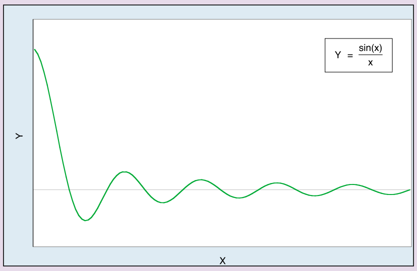


Figure F: The sinc function.

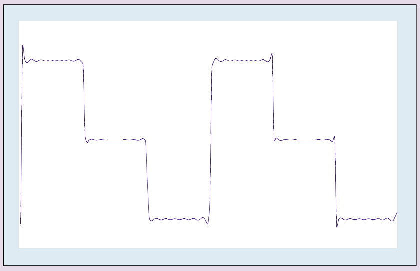


Figure H(i): The waveform generated by the spectrum in Figure H(ii).

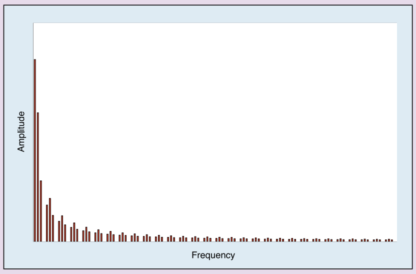


Figure D(ii): The harmonics that produced the wave in figure D(i).

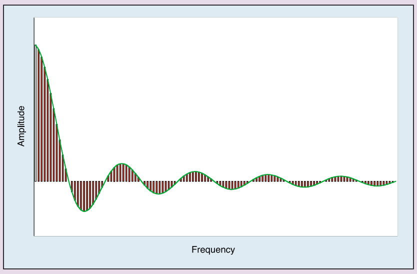


Figure G: How the sinc function determines the amplitudes of pulse waves' harmonics.

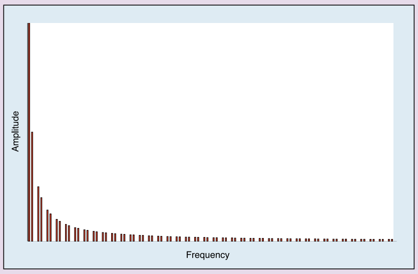


Figure H(ii): A 1/n spectrum with every third harmonic missing.

can now mix this with the first. Take advantage of the JX10's ability to detune Patches against one another, add a bit of external reverb and... Wow! The results can be gorgeous; rich, involving, and with impressive depth. What's more, we *still* haven't even taken advantage of the JX10's two built-in chorus units! Every note you hear is simply four detuned and modulated oscillators shaped by two low-pass filters and a pair of VCAs.

But now it's time to create our ultimate ensemble sound... just change parameter 64 from 'OFF' to '1' in both Patches to apply chorus to each. You'll find that the resulting sound is far better than what you can coax from the majority of string machines.

### Epilogue

So there we have it... we've created superb vintage-style 'string' and 'ensemble' sounds

## A Different Way Of Looking At PWM

As explained at the start of this article, the effect of PWM can be recreated even on synths that don't offer it. This is made possible by considering PWM as the product of two pitches. I hinted at this last month, but it's now time to explain it rather more thoroughly.

Let's start with the square wave shown at the top of Figure Q below. As you can see, the waveform is static — in other words, the waves' shapes do not change as time passes. But this need not be the case — there are many ways in which we can alter them. We could smooth one or more of the sharp corners off the square wave, make the vertical bits slope (creating a trapezoid waveform), put some wiggles in the horizontal bits, or create combinations of these... and so on. Sure, the resulting waveforms might be difficult or even impossible to generate using analogue electronics, but they are nonetheless valid, and an appropriately programmed digital synth should be able to generate any of them with ease. Nevertheless, one modification to the wave shape of a pulse wave that many analogue synths can perform is modulation of the duty cycle.

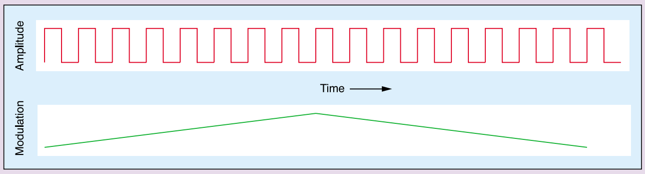


Figure Q: A pulse wave with a duty cycle of 50 percent, and a triangle-wave modulator.

Looking now at all of Figure Q, you can see a second, low-frequency modulating waveform shown in green underneath the square wave. I have chosen a triangle wave for the low-frequency modulator, because this will best demonstrate the principles that I wish to describe.

Let's imagine that when the modulator is at its minimum voltage, the duty cycle of the audio wave is very close to 0 percent. This means that the first pulse shown in the audio waveform has almost zero duration. As the modulator increases in voltage, the duty cycle increases until, when it reaches its maximum, the next pulse has a duty cycle of virtually 100 percent. As the modulation voltage falls, the duty cycle decreases until the modulator reaches its minimum once more, after which point the whole cycle begins anew. I have shown this as Figure R below.

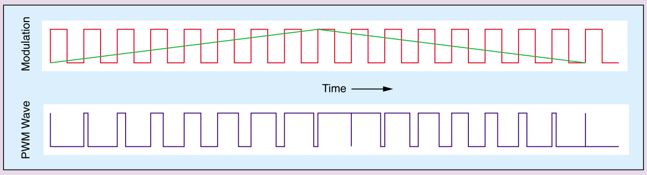


Figure R: How a modulator affects a pulse wave to generate PWM.

As we've seen in the box on the nature of the pulse waveform (on the previous page) the duty cycle of a pulse waveform is intimately related to its harmonic content. This means that because the duty cycle in the PWM waveform is constantly changing, so too is the harmonic content of the wave. But, as I stated last month, it is not easy to see why this alone should give PWM its rich, chorused sound. Sure, analysing the harmonic spectrum provides us with a powerful way to describe and understand what's happening to the signal, but it's not necessarily the best way to explain *why* we hear *what* we hear. We need to find a better one.

When we consider a signal, there are many ways that we can describe it and analyse it. The most common is the amplitude/time graph, but I've already used a second type this month, the *frequency*/time graph, which allows us to see how the spectral content of the signal changes in time, and now I'm going to introduce a third, the 'differential' graph, which shows how the gradient of the signal in the amplitude/time graph changes over time. If you're familiar with maths, you'll recognise this concept, but don't worry if you're not, as I'm now going to attempt to differentiate the PWM signal using words to describe the results, with no maths whatsoever. I can do this because of the simple nature of pulse, triangle and sawtooth waves, but it would be almost impossible if we were to attempt this for more complex waveforms.

Let's start at the left-hand edge of the modulated audio signal in Figure R. We're going to scan along it from left to right, and every time we encounter an upward transition in the waveform, we're going to draw an upward arrow at that position. Likewise, every time we encounter a downward transition, we're going to draw a downward arrow.

OK, I'll admit that the first pulse is a bit confusing, because it looks like we should draw a single downward arrow, but this is misleading. This is the pulse with a duty cycle of almost 0 percent, so we must draw both an upward arrow *and* a downward arrow at that position. Moving to the right, it then becomes obvious that there is an 'up', a 'down', an 'up', another 'down'... and so on, until we come to another pulse where the 'down' and 'up' arrows are almost at the same position (the pulse with a duty cycle of almost 100 percent). This is then followed by a 'down', an 'up', a 'down'... and so on (again) until we reach the next zero-percent pulse. If you study Figure S (above), you can see that — although it's a bit complex to describe in words — what we've done is visually very clear and intuitive.

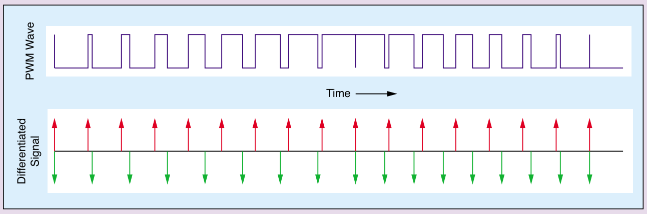


Figure S: Looking at the PWM signal in a different way.

I'm now going to split Figure S into two separate graphs: one that comprises the first half of the lower part of Figure S, in which the down arrows are more widely spaced than the up arrows; and the other comprising the second half, in which the converse is true (see Figures T and V, right and above right). If you refer back to Figures Q and R, you can see that these two new graphs are, respectively, the regions in time during which the modulating signal is rising, and when it is falling.

Now comes the bit that is somewhat less than intuitive... Despite the fact that they are derived from a single waveform, we can treat the 'up' arrows in Figure T as a single signal,

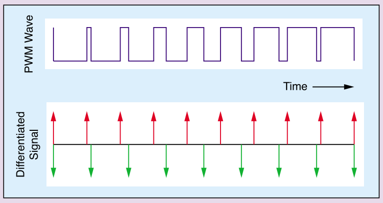


Figure T: Describing the PWM signal when the modulator is rising.

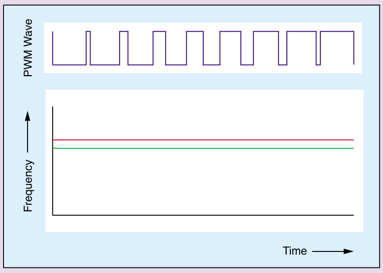


Figure U: The two signal components that comprise the first part of the PWM wave.

on two unlikely instruments; one a hybrid analogue/digital synth with DCOs, and the other an entirely digital synth with nary a voltage-controlled wotsit in sight. Neither offers PWM, but both can sound as rich, lush, and warm as many more highly rated (and considerably more expensive) instruments.

In short, if you're thinking of shelling out several hundred pounds on some unjustifiably revered string synth... *stop.* Don't be a fashion victim! My advice would be to learn how to program a JX10, and save yourself a few hundred quid.

**I would like to thanks Dr Christopher Hicks of CEDAR Audio for his invaluable time and patience as he helped me to avoid the unintentional application of signal-processing theory egg to my face.**

**and the 'down' arrows as a second, independent signal. We can do this because the sizes of the arrows remain constant. What's more, because the spacing of the arrows comprising each signal is also constant, we can infer that the frequencies of these signals are also constant — for the duration represented by Figure T, at any rate. At this point, we have no idea what the shapes of the separate 'up' and 'down' signals might be. What we do know, however, is that the 'up' signal completes nine cycles (the red arrows) in the same time that the 'down' signal completes eight cycles. We can therefore state with confidence that there are two audio signals present, each of constant frequency, but with one lying at nine-eighths of the frequency of the other. I have shown this as Figure U (below left on the previous page).**

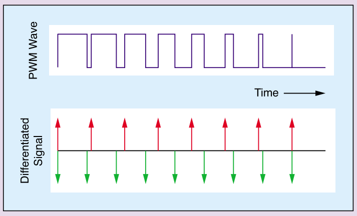


Figure V: Describing the PWM signal when the modulator is falling.

**Looking now at the part of the PWM wave generated when the modulator is *falling* (see Figure V, above left), we see that there are seven complete 'up' cycles for every eight 'down' cycles. We can therefore state that there are again two signals present, with one having a frequency of eight-sevenths of the other. I have drawn this as Figure W (left). Note that this shows how the signal with the higher frequency in Figure U — the red 'up' signal — is the one which has the lower frequency in Figure W.**

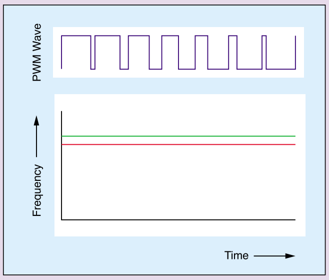


Figure W: The two signal components that comprise the second part of the PWM wave.

**Next, putting Figures U and W together, I can represent the frequencies of both audio signals in the PWM cycle, as shown in Figure X (below). Note that I have also compressed the timeline by a factor of four to generate Figure X, so that it shows four complete PWM cycles. This helps to bring out the useful factors in this analysis, which are less obvious when viewing just one cycle.**

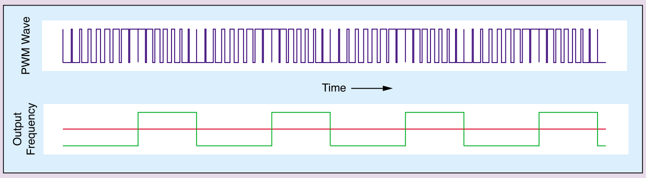


Figure X: The frequencies of the two components that comprise the PWM waveform.

**Now we can see what is happening... As already stated, in this way of looking at things, the PWM waveform comprises not one, but *two* independent audio signals, one of constant frequency (the red line that represents the 'up' signal) and one that is undergoing pitch modulation above and below this frequency (the green line that represents the 'down' signal). Figure X also makes it clear why the PWM wave sounds so rich. In having two signal components, the pitch of one of which is frequency-modulated with respect to the other, the PWM wave produced by a single oscillator is no different from the signal produced by two independent oscillators, the pitch of one of which is, umm... frequency-modulated with respect to the other.**

**But this isn't the end of the matter. We have considered only the case in which the leading edges of the pulses are unaffected by the modulator. In other words, the leading edges (ie. the 'up' signal) are always equally spaced, no matter what else is happening to the waveform. But it is just as valid to consider the case in which both the leading edges *and* the trailing edges move with respect to the centre of the top of the pulse. If they are affected equally, it doesn't take too much of a leap of faith (or too much graph paper) to realise that the frequencies of both signal components will now be altered, as shown in Figure Y (above).**

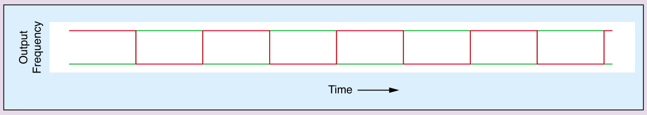


Figure Y: The frequencies of the components when the leading and trailing edges of the pulse are affected equally.

**Now that we have shown PWM to be the sum of two signals, at least one of which is frequency-modulated with respect to the other, it would seem reasonable to speculate that we can recreate it by determining the natures of the 'up' and 'down' signals themselves, and combining them in the correct fashion (mathematically, this combination process is the opposite of differentiation, and is known as *integration*, but you needn't worry about that if you don't want to). So what about the nature of the 'up' and 'down' signals? Well, imagine drawing the two waveforms you would get by responding independently to the 'up' and 'down' impulses in Figure S (remember that we can treat the 'up' impulses as one signal, and the 'down' impulses as a second, independent signal).**

**Again, this sounds tricky, but Figure Z (right) should make everything clear. As you can see, we obtain two 'staircase' waves, one ascending to infinity at a rate determined by the gaps between the 'up' arrows, and the other descending to minus infinity as defined by the 'down' arrows. I'll admit that these are not conventional-looking synth waveforms, and they are not cyclic in any traditional sense, but this does not preclude their validity. In fact, if you take a piece of graph paper and add them together, you'll find that you obtain a perfect PWM waveform.**

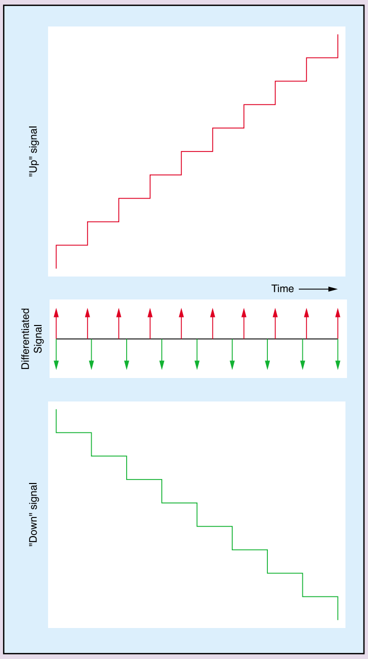


Figure Z: The nature of the separate 'up' and 'down' signals.

**But hang on a moment... If one signal climbs ever on upward, and the other descends downward forever, surely there can be no way for us to construct a true PWM signal using oscillators that do not have the ability to modulate the pulse width directly? After all, synth oscillators simply do not have infinite voltage ranges.**

**Nevertheless, it can be done — and a look at the main text of this month's article will tell you how.**

# Synth Secrets

## Synthesizing Bowed Strings • The Violin Family

Gordon Reid

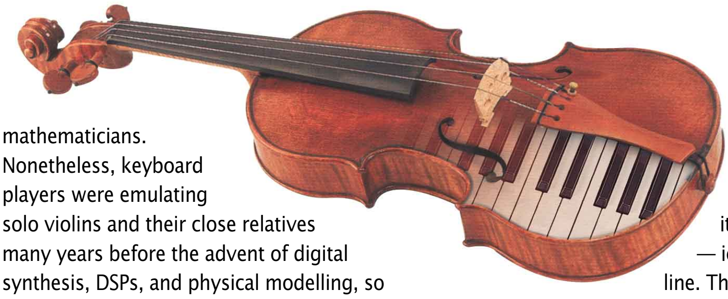
**Following our success at synthesizing the sound of analogue string *machines*, we hone our techniques with a view to recreating the sound of the real thing...**

For the past few months, we've been using synthesizers to imitate a specialised form of electronic keyboard: the string synth or string machine. Designed to generate a sound reminiscent of ensemble strings, and to appeal to players who could not afford a Mellotron, this was the instrument first developed in the 1960s by Ken Freeman and later perfected by Italian organ and accordion manufacturers such as Logan.

Nobody in their right mind would claim that string synths were indistinguishable from the massed strings of a real orchestra, but they proved to be adept at creating the impression of such things and, when played sympathetically, could almost fool you. Given that we used sawtooth waves to generate our ensemble sounds, it would seem reasonable to infer that, if multiple sawtooth waves imitate the sound of multiple string instruments, one sawtooth might imitate the sound of a *single* string instrument. It's reasonable... and it's correct. So, this month, we're going to study bowed string instruments and learn why — as is fairly well known — we use sawtooth waves to synthesize the sounds of violins, violas and cellos.

### Strings & Bow

The violin is the most studied of all the classical instruments. Numerous books analyse and explain its shape, its construction, the materials used, the nature of the strings, the effect of the bridge, the bow, the bowing position, the differences introduced by fast and slow bowing, the effects of higher and lower bowing pressures, the effects of vibrato, pizzicato... and many other factors. In addition, there are scores of scientific papers, many of which discuss the physical modelling of these attributes. If you've ever tried to read these papers, you'll never be rude about a Yamaha VL7 or a Korg Prophecy again; modelling a bowed string instrument is the work of *magicians*, not mathematicians.



Nonetheless, keyboard players were emulating solo violins and their close relatives many years before the advent of digital synthesis, DSPs, and physical modelling, so there must be a way to achieve acceptable results on subtractive synthesizers. So let's start — as we have in the past — by looking at the waveform produced by the excitation mechanism of the instrument; in this case, the string and the bow.

Figure 1 (below) shows a simplified view of a violin string and bow. The string itself is stretched taut between two anchor points, and a bow of stretched horsehair is placed upon it and then dragged in a perpendicular direction across it.

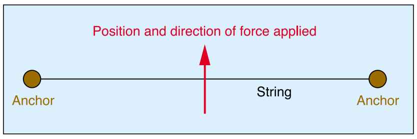


Figure 1: A simplified representation of a string and bow.

Because of friction, the bow applies a displacing force to the point of contact on the string, which is stretched progressively into the shape shown in Figure 2 (below). As this happens, the string's tension generates a restoring force that tries to return it to its rest position — ie. a straight line. This force increases as the displacement increases so, at some point, the restoring force exceeds the ability of friction to drag the string further. At this point, the string begins to slip...

The amount of friction between two objects moving with respect to one another is less than that between two objects that are stationary with respect to each other, and we call these two types of friction 'dynamic' and 'static' respectively. Think about it: a car parked by static friction on a steep hill will tend to remain stationary, but if it does start to slide, the lesser dynamic friction between it and the road will be insufficient to arrest its motion until the incline flattens out (or something more dramatic occurs on the way down!). Relating this idea to the bowed string, static friction will 'stick' the string to the bow until the returning force causes it to slip. At this point, the interaction between the bow and the string becomes one of dynamic friction, which is the lesser force, so the string will snap back. However, it does so to a point *beyond* the original rest position because, just like a plucked string, its momentum takes it

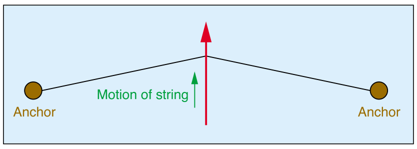


Figure 2: The bow dragging the string.

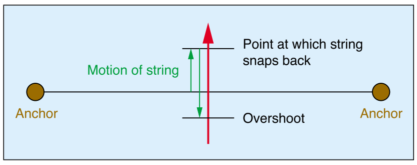


Figure 3: The 'snap back' of the string.

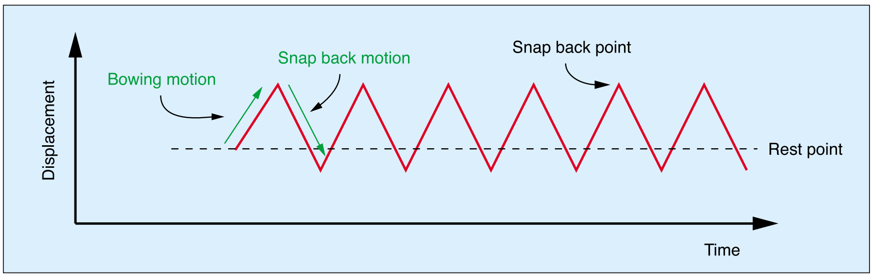
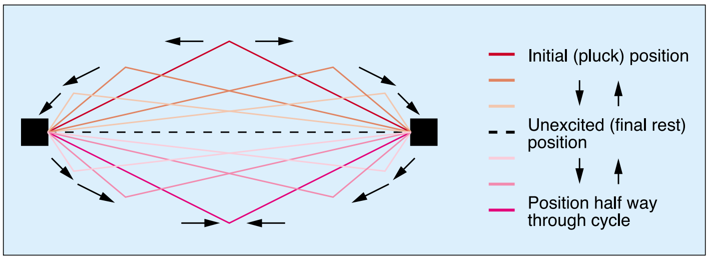


Figure 4: The displacement diagram for the bowed point on the string.



Initial (pluck) position
Unexcited (final rest) position
Position half way through cycle

Figure 5: How two pulses moving in opposite directions produce the wave motion of the plucked string.

into the region where its tension slows it to a halt (see Figure 3, below left).

Once the string has reached a point somewhere beyond its rest position, the dynamic friction is great enough to stick the string to the bow for a second time. At this moment, static friction takes over, and the string is displaced again until it reaches the 'snap-back' point, and the cycle repeats.

Surprisingly, if the bowing position is in the centre of the string, the speed at which the string 'snaps back' appears to be much the same as that as which it is dragged. We can draw this motion as shown in Figure 4 above: it's a triangle motion, although it's one whose centre is offset from the rest position of the string.

Now, it's tempting to think that because the bowed point describes a triangle wave, the audio waveform generated by the string is also a triangle wave. As I showed when I analysed the plucked string, this is sometimes a reasonable conclusion... but on this occasion, it's wrong.

If you refer back to the part of this series in which I analysed the behaviour of a plucked string (see *SOS* August 2001, or surf to www.sound-on-sound.com/sos/aug01/articles/synthsecrets28.asp), you'll recall that two waves — one travelling left to right, the other right to left — combine to produce the wave motion of a plucked string (see Figure 5 above).

But a bowed string is not the same as a plucked string, and the physics of its vibrations are different, because the string does not oscillate freely. Without going into the maths to prove why this should be so, there appears to be just one wave in the bowed string, and this travels around as shown in Figure 6, right. The speed with which this wave travels, and therefore its frequency, is the same as that of the freely vibrating string, and although this might seem to be a coincidence, it must be so, or it would not be possible to play pizzicato notes on a violin at the same pitch as you obtain when bowing it.

If you study Figure 6 more carefully, you can see that, as the wave travels around the string, the point under the bow describes the triangular motion in Figure 4. This would seem to imply that the bowing speed determines the pitch of the note thus created. But we *know* that this is wrong; any violin player will tell you that the musical pitch is not determined by the bowing speed. So what changes as you move the bow faster or more slowly? It's the point at which the string snaps back, which is, if you think about it, the distance from the tip of the triangle to the string. Therefore, when the player moves the bow with greater velocity, the *amplitude* of the wave increases, but the frequency remains constant (see Figure 7, right).

Now we can consider what happens if the player bows the string in different positions.

## About Rosin

**The coefficient of friction of horsehair alone is too low to excite strings sufficiently to play a bowed instrument, so players cover their bows with a substance called rosin, a gum that becomes sticky just above room temperature. When warmed by the friction of the bow dragging across the string, this provides the necessary friction to pull the string out of shape. What's more, in a delicious twist of fate, the static friction of rosin rises as it gets warmer, so that it is more efficient at displacing the string during the dragging period. At the same time, its dynamic friction decreases, so the string snaps back even more smoothly than it otherwise could. Perfect!**

We already know what happens when it is bowed in the centre: the bowing point moves in a triangle wave, and there are no even harmonics. This is because, as with the plucked string, all the even harmonics would require the centre of the string to be at rest, which is impossible when that is being bowed (if this explanation has lost you, it's explained in much more detail in that August 2001 instalment of this series, the one about the behaviour of the plucked string). It's possible to deduce that if the bow is one third of the distance from the bridge to the nut there can be no third, sixth, ninth, and other 'third' harmonics in the waveform. Similarly, a bowing position a quarter of the way from the bridge eliminates the fourth, eighth, 12th (and so on) harmonics. But if the bowing position is some arbitrary distance from the bridge, it is likely that *all* the harmonics will be excited to a greater or lesser degree. This proves to be the case, but it still does not tell us what waveform the instrument produces.

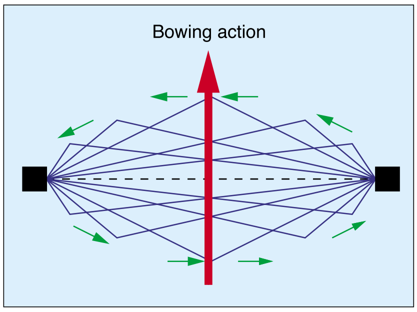


Figure 6: The wave motion of a bowed string.

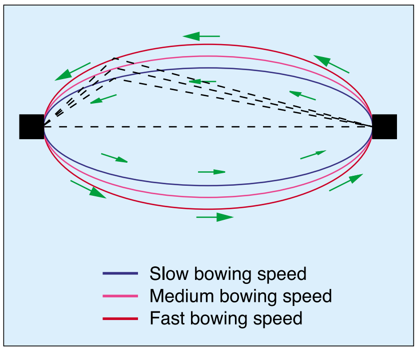


Figure 7: The bowing speed affects the amplitude, not the frequency of the note.

To determine this, we need to consider something else — the forces acting upon the bridge which separates the strings from the body of the instrument.

## The Forces On The Bridge

Consider the following two ideas:

- While the bow is displacing the string, the resulting forces strain the bridge and pull it out of position;

- While the string flies back, the bridge

▶

returns progressively to its rest state.

This would suggest that some sort of sawtooth wave is invoked in the bridge. And so it proves to be; scientists have observed the forces applied by the strings upon the bridges of bowed instruments, and found that they are remarkably well represented by sawtooth waves (see Figure 8, below).

Other waveforms are obtained when the bowing is applied differently, or inappropriately. For example, Figure 9 (below) shows what happens when the player fails to press the bow hard enough onto the string, allowing it to slip twice in each cycle. This 'double-slip' motion does not change the pitch, but more often creates a new tone that violinists call 'surface sound'. If they had ever studied hard sync on an analogue synth, they would understand what they were hearing!

Even more extreme is the waveform resulting from *multiple* slips of the bow; this creates tones that — outside of avant-garde playing — are best avoided by skilled players (see Figure 10, at the bottom of the page).

Despite the apparent completeness of this analysis, many secondary factors affect the waveform induced in the bridge. For example, the string is not a perfect oscillator with infinitely sharp cusps at the nut, bridge and bowing point, nor is the bow infinitely narrow, and these factors have a number of consequences. Firstly, the pitch of the note goes slightly flat as it becomes louder. Secondly, there is jitter in the pitch as the 'corner' of the wave in Figure 6 passes under the bow. If we were being thorough, we would also have to consider the reflection of waves at the point of string/bow contact, and the effect that these have as they bounce around in the two sections of string either side of the bow. But we won't (phew!!).

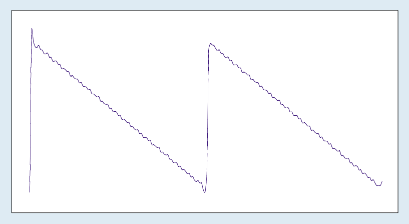


Figure 8: The force waveform measured at the bridge of a violin.

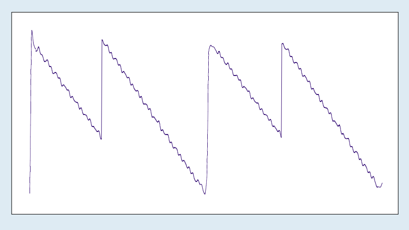


Figure 9: The waveform resulting from 'double-slip' motion caused by insufficient bowing pressure.

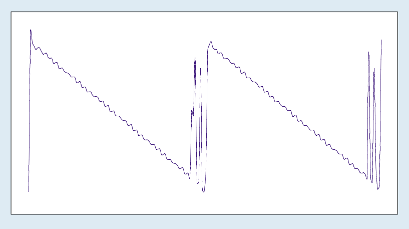


Figure 10: A waveform resulting from 'multiple fly-backs'.

We could also consider how players obtain different sounds by tilting the bow to increase or lessen the amount of hair in contact with the strings, and what happens if the player does not drag the bow perpendicularly across the strings. Actually, we've all heard the result of the latter: it's the ghastly squeal of helpless, dying felines, murdered by the enharmonic, longitudinal vibrations excited by novice violinists.

## The Vibrating Body

Like the guitars that we studied when investigating the plucked string way back in 2001, the timbre and playability of bowed instruments are largely determined by the properties and motions of their bodies — the top and bottom plates, the air between them and, to a lesser extent the sides, neck, and other bits and pieces. It turns out that these motions are even more difficult to model than those of the guitar, in part because there is a post in a bowed instrument that links the top and bottom plates.

This complex situation is not aided by the fact that just as on a guitar or piano, the strings of a bowed instrument interact with the body and each other, absorbing and releasing energy in complex ways that are far beyond the scope of this explanation. Nonetheless, the overall shape of the modes is quite similar from instrument to instrument within a given family. This is as it must be, or it would not be possible to recognise all violins as violins, all cellos as cellos, and so on.

Figure 11 (above right) shows a simplified representation of the low-to-mid end of a violin body response. As you can see, this is dominated by some prominent modes produced by the top and bottom plates of the instrument. If the diagram looks familiar, you shouldn't be too surprised. I drew a similar one when I analysed the acoustic guitar and, although the positions of the modes are different, the broad shape is comparable.

Despite these similarities, your ear will never mistake a Stradivarius for an Epiphone and, if you could observe the modal vibrations of each, you would quickly see (as well as hear) the differences between them. Figure 12 (above) shows one of the simpler (!) modes of the bottom plate of a violin, and this is quite different from the equivalent guitar mode.

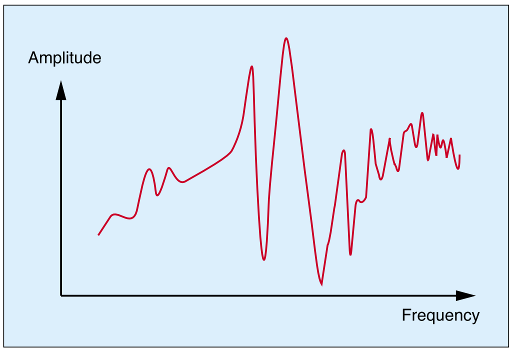


Figure 11: A simplified representation of the low and mid-frequency response of a violin.

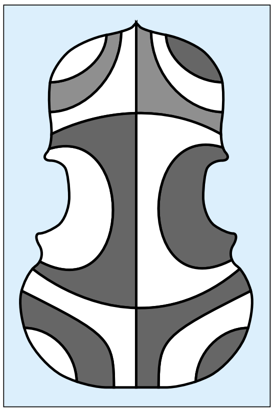


Figure 12: One of the simplest violin modes!

Now, you might be tempted to think that if we measured the frequencies of a given instrument's modes, we could build a filter bank that would recreate the response of its body, and that passing a sawtooth wave through this filter would then create an accurate imitation of the original tone. Yes? No... not quite.

Things are never as simple as we would like, and every bottom plate mode interacts

## The Bow

**Throughout this article, I have overlooked the characteristics of the bow itself. However, when you think about it, the bow provides far more than just a means to energise the strings. The wood is under tension and has resonant frequencies, as must the tensioned horsehair, and the combination of the two must interact in complex ways, influencing the way in which the string responds to the player's technique. For example, the 'bounciness' of the bow and its ability to resume its shape will determine how string players create playing styles such as spiccato and sautille. Hmm... perhaps there are good reasons why some bows are worth thousands of pounds, while you can find others for just a few quid.**

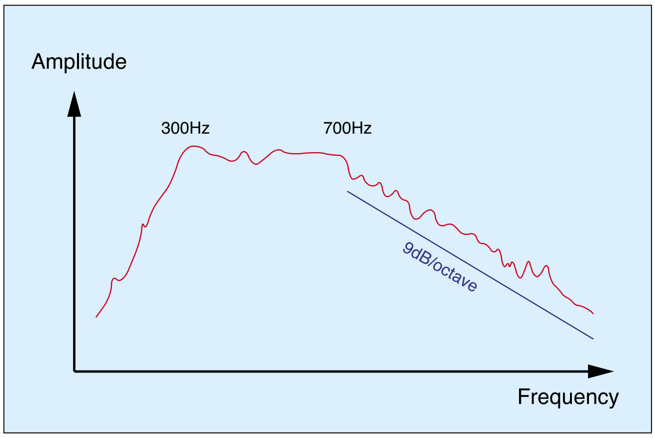


Figure 13: A measurement of a violin body's spectrum when energised by a sine-wave oscillator.

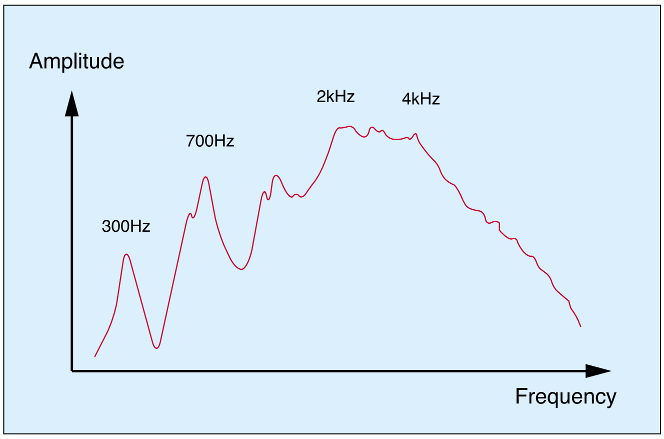


Figure 14: A measurement of a violin's spectrum when played with the bow.

with dissimilar modes of the upper plate (which is affected by the 'f' holes present in all of the violin family), and both are affected by the modes of vibration of the air partially trapped between them, and by the sound-post inside the body. Oh yes... and unlike a guitar, the bridge of a bowed instrument has a complex resonant response of its own, with modes in all three dimensions. And this interacts with all the body modes, and the neck, and it passes energy back to the strings, and... Ouch! This is a recipe for a synthesis headache just as nasty as the one I got from... well, synthesizing a plucked-string response.

Ignoring these complicating factors for a moment, is it possible to isolate an instrument's body from the neck, bridge and strings, and measure its frequency response? Fortunately for the purposes of this exercise, it is; academics achieve this by suspending the body and then energising it at every frequency using a swept sine-wave oscillator. They then measure the radiated sound. In the case of the violin, the observed result is a curve with a flat response across a few hundred Hertz, a steep roll-off in the bass, and a gentler roll-off of about 9dB per octave in the upper-mid and high frequencies (see Figure 13, above left).

However, this information isn't much use, because a bowed instrument's body is *not* energised by a single frequency — it is energised by the energy transmitted by the bridge. This, as we have already seen, starts its life as a sawtooth wave *rich* in harmonics, and is then further shaped by the resonant response of the bridge itself. This harmonically rich energy interacts with the body to generate a different response, and it is this that we must recreate if we wish to synthesize a bowed instrument accurately (see Figure 14, above).

This is still not the end of the story, because experience tells us that the sound of, say, a violin or cello is different when heard from in front or behind, above or below. Again, this makes sense... were it not so, it would be simple to set up the microphones to record the instrument, position would be irrelevant, and everybody would be a competent sound recordist. Clearly, this is not the case!

The results obtained from measuring the sound dispersion of bowed instruments at various positions leads to complex patterns. For example, experiments show that cellos radiate more energy forward at 200Hz, but more backward at 250Hz, and more upward at 800Hz. So, if you place three microphones near a cello — one in front of, one behind, and one above the instrument — you will obtain different timbres (although all will still be recognisable as sounds produced by a cello). This means that, if we want to synthesize a bowed instrument in a truly accurate fashion, we should use some sort of surround-sound processor to imitate the differences that you will hear when you move around in its soundfield. However, given my final comments (see below), and the fact that most of us would be overjoyed if we could synthesize even the tone of a badly miked-up cello from scratch, we can probably omit this step.

## Wolf Tones

**Violins, violas and, in particular, cellos, suffer from the existence of so-called 'wolf tones' — notes that are difficult to produce on a given string because the coupling of the body resonances, bridge modes, and string vibrations cause the string's energy to pass from the string to the body and back again at a frequency of a few Hertz. This creates a flutter that makes it very difficult to sustain the note correctly.**

## Synthesizing Bowed Instruments

Perhaps luckily for this analysis, understanding bowed instruments is not trivial. Even today, no-one can tell you precisely why a 'Strad' sounds superior to a well-crafted modern violin, even when the former may have undergone significant modifications over the past couple of hundred years.

For this reason, when we try to synthesize the violin, viola or cello, it is more fruitful to concentrate on the broad nature of the sound than it is to become over-concerned with details. I have broken this down into three components, as follows:

- Given that the bodies of guitars and bowed instruments are energised by sawtooth waves, it is vital to imitate the 'shape' of the sound generated by the bowing action, and to differentiate it from the shape of a plucked string (there are good reasons why pizzicato played on a violin or viola shares many of the sonic attributes of a banjo!).

- Secondly, the shape of the spectrum in Figure 14 is important. The timbre of a violin is strongly linked to the dominant body resonances in the region of a few hundred Hertz, as well as the broad combination of resonances in the region of 2kHz to 4-5kHz or thereabouts. Without these (or their equivalents for the viola or cello) the sound will not be realistic.

- Finally, we should take account of the way in which the player plays the instrument. An Irish fiddler produces a sound very different from that of a classical soloist, and Stephane Grappelli produced a sound that was markedly different from either. Perhaps the two most easily synthesized performance characteristics are glide and vibrato, so we should concentrate on imitating these as accurately as possible.

But, for now, we've run out of space, so we'll have to address these issues next time. SOS

# Synth Secrets

Gordon Reid

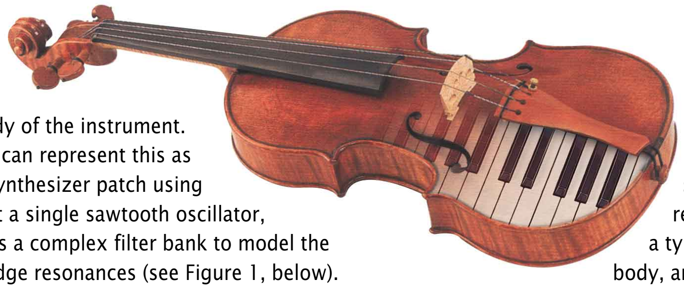
## Practical Bowed-string Synthesis

**Having looked at the mechanics of how a bowed string instrument generates its sound last month, it's time to put these principles into practice, using nothing more complex than a miniKorg 700 monophonic synth...**

Last month, we discussed the nature of the orchestral instruments that are usually bowed to create their sounds. These are commonly accepted to be the violin, viola and cello, although — despite its use as a plucked instrument in jazz and rock & roll — the double bass is also a member of the family. We will now attempt to recreate the sounds of the smallest of these using a simple, analogue, subtractive synthesizer.

### The Basics

As I explained in last month's instalment of this series (see www.sospubs.co.uk/sos/apr03/articles/synthsecrets48.asp), the action of the bow upon the string and the action of the string upon the bridge produce a sawtooth wave in the bridge itself. This energy is then modified by the bridge resonances before being transmitted to the body of the instrument. We can represent this as a synthesizer patch using just a single sawtooth oscillator, plus a complex filter bank to model the bridge resonances (see Figure 1, below).



What happens when we energise the body of the instrument is a little more complex, although we can describe it in similar fashion; the body has a complex frequency response that imposes another set of resonances and anti-resonances upon the sound.

Figure 2(a) (below left) shows the response for a typical violin body, and Figure 2(b) shows the rather different spectrum obtained when the body is energised by the modified sawtooth wave produced by the filter bank in Figure 1. As you can see, there are three prominent resonant regions in Figure 2(b), with a sharp roll-off in the bass, and a gentler roll-off at high frequencies. And, complex though this appears to be, we can approximate it using just three synthesizer modules: a low-pass filter, a high-pass filter, and a resonant filter bank (see Figure 3, above right).

Fortunately, when modelling the violin,

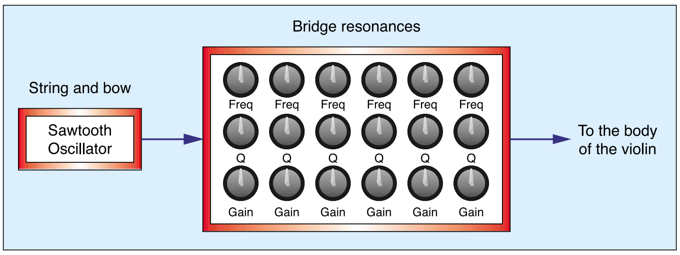


Figure 1: Modelling the bow, string and bridge.

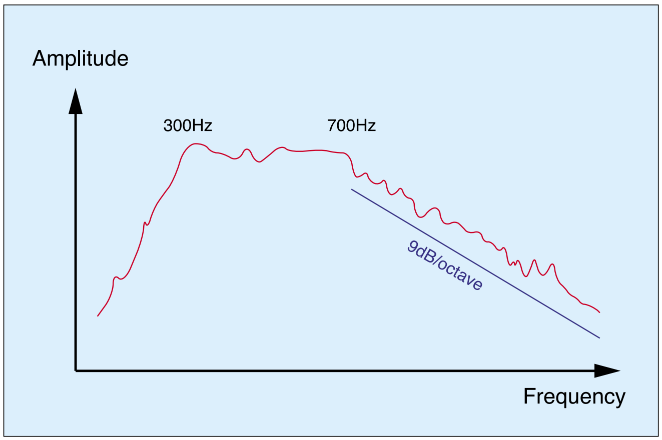


Figure 2(a): The frequency response of a violin body.

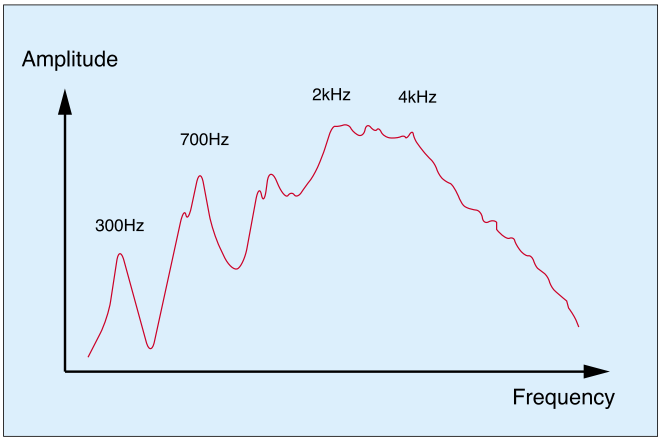


Figure 2(b): The frequency response obtained when energising the violin body using a string, bow, and bridge.

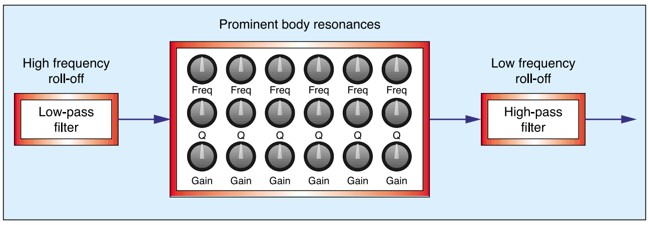


Figure 3: Modelling the combined frequency response of the violin's elements.

we don't need to add the filter bank in Figure 1 to the filter bank in Figure 3. We can use a single bank and set the filter frequencies, gains and Qs to the appropriate values for the combined response of the bridge and body. This then gives us a model for the basic timbre of a violin's sound, as shown in Figure 4 below.

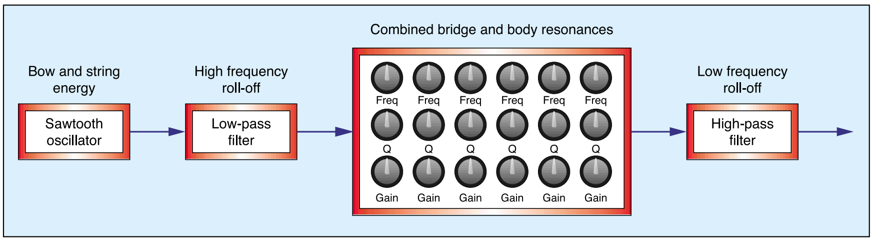


Figure 4: A simple model of the initial violin timbre.

Now, I'm at a loss to recall an integrated subtractive synthesizer that offers a low-pass filter, a high-pass filter and a resonant, parametric filter bank. However, this need not be a problem, because if we ignore a number of secondary effects, the order in which the filter appears need not matter to us, and we can place the filter bank *after* the low-pass and high-pass filters. In other words, we can create the basic sound using any synth with a low-pass filter and an high-pass filter, and then pass the output through an *external* filter bank. Of course, not everybody has access to an external filter bank, but there are a number of affordable units available. I have one of them... an Analogue Systems RS360 Vocal/Phase Filter Bank (as shown in Figure 5 below), so I will use this to replace the signal model in Figure 4 with that in Figure 6. By the way, note that Formant synthesis (for this is what we are discussing — see the box above) requires at least three formants per sound. This means that the RS360 or an equivalent is the minimum suitable configuration for our purposes.

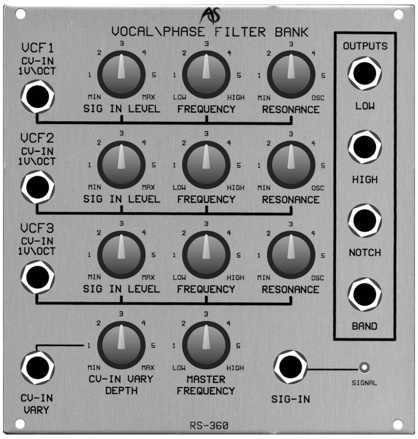


Figure 5: The Analogue Systems RS360 filter bank.

If you study Figure 5, you will see that each of the RS350's three filters offers a Sig In Level, a Frequency control, and Resonance. Each filter produces the four common filter characteristics (24dB-per-octave low-pass, 24dB-per-octave high-pass, 12dB-per-octave band-pass and 12dB-per-octave notch filtering), and the cutoff frequencies of the high-pass and low-pass filters are the centre frequencies of the band-pass and notch filters. The summed outputs on the right-hand edge of the module offer the low-pass outputs of all three filters, the high-pass outputs of all three, and so on.

Now, we're not interested in the low-pass and high-pass filters offered, nor are the notch filters of any use this month. But if we use the summed outputs of the band-pass filters, we can achieve some useful results.

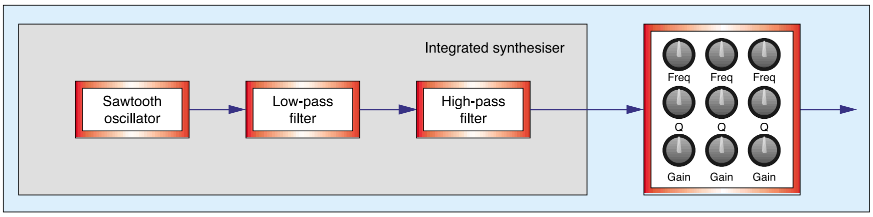


Figure 6: Combining an integrated synth and an external filter bank.

## Band-pass Filter Banks

**Filter banks of the sort described here are often used to create Formants. These are the static resonances inherent in sound generators such as the human voice and hollow-bodied musical instruments, and they are important elements within sound synthesis. This is because, just as the precise positions and shapes of the formants in a human voice allow you to identify the vowel sounds spoken, they make the timbres of acoustic instruments consistent and recognisable from one instrument to the next. It therefore follows that recreating the formants of a given instrument represents a big step toward the accurate synthesis of its sounds.**

To set up the RS360 appropriately, we will set the first two filters to accentuate the body resonances at 300Hz and 700Hz. We do this by setting the Frequency controls of VCF1 and VCF2 to the appropriate positions, setting the signal levels to imitate the height of the peaks in Figure 2, and then adjusting their resonances to emphasise any signal partials that lie at, or close to, these frequencies. To create the broader peak at 2kHz to 4kHz, we set VCF3 to about 3kHz and use a lower resonance. This allows a wider spread of frequencies to pass.

I have summarised the settings necessary to recreate Figure 2 in the table below, and I can represent their combined frequency response as Figure 7 (shown on the next page).

| FILTER | SIG IN LEVEL | FREQUENCY | RESONANCE |
|---|---|---|---|
| VCF1 | 4 | 300Hz | 3.5 |
| VCF2 | 5 | 700Hz | 3.5 |
| VCF3 | MAX | 3kHz | 2 |

At this point, you may be wondering how on earth you go about tuning a filter bank to precise frequencies. Synthesizer filters with calibrated initial cutoff frequencies are vanishingly rare, and Analogue Systems' annotation from 'Min' to 'Max' is of no help whatsoever. So here's the trick...

Placing a suitable attenuator somewhere in the signal chain between the filter bank and your speakers, you should set the cutoff of VCF1 to a middling value, and then increase the resonance to maximum. A tortured shriek will result... and it will be you doing the shrieking if you ignored my warning about the attenuator. The filter is now emitting a sine wave at the cutoff frequency so, given a suitable reference, we can tune it to the desired pitch.

Few of us have a pitch-to-frequency chart in front of us (and even fewer have

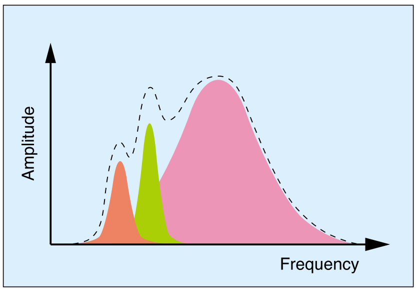


Figure 7: Modelling the violin frequency response of Figure 2.

memorised one) so we need to be able to judge the cutoff frequency using a simple, mnemonic rule of thumb. The one I devised and use is this: We know that the fundamental frequency of the 'A' above middle 'C' is 440Hz or thereabouts. We also know that the 'A' above this oscillates at double the frequency, and the 'A' below oscillates at half the frequency. Furthermore, we know that a perfect fifth — ie. close to the 'E' above A440 — oscillates at 3/2 times A440, as do all the other 'E's relative to the 'A's below them. We can, therefore, create a simple table that defines a number of frequencies across the keyboard, as shown here.

| NOTE | APPROXIMATE FREQUENCY (HZ) |
|---|---|
| A0 | 55 |
| E1 | 82.5 |
| A1 | 110 |
| E2 | 165 |
| A2 | 220 |
| E3 | 330 |
| A440 (MIDI A3) | 440 |
| E4 | 660 |
| A4 | 880 |
| E5 | 1320 |
| A5 | 1760 |
| E6 | 2640 |
| A6 | 3520 |

Tuning our filters now becomes simple. If you want to set VCF1 to about 300Hz, you find the closest Figure in the table (which is E3 at 330Hz) and — by simple comparison — tune the self-oscillation to a note a semitone or two below this; ie. in the region of D#3 or D3. Likewise, the closest to 700Hz — the frequency we desire for VCF2 — is E4 at 660Hz, so you tune the self-oscillation of VCF2 to F4 or thereabouts. Finally, 3000Hz lies about halfway between E6 and A6, so you tune VCF3 in the region of F#6 or G6. Having found the correct setting for each filter in turn, you then turn down the resonance to the values shown in the table on the last page (a self-oscillating filter is not what we want in our sound this month, after all). It's a crude method, given that pitch operates on a logarithmic scale, not a linear one — but it works.

## Inserting The Waveform

Now we have to select a synth to provide the filtered sawtooth wave that we're going to pass through the RS360. Oh yes... and we're going to need an amplitude envelope, an LFO and a keyboard or other controller of some sort.

These are simple requirements, and we could use almost any synth to satisfy them. I'm going to turn to the first synth I ever owned; one of the most basic ever built, yet still capable of producing some rather super sounds. It's the Korg 700.

Figure 8 (below) shows the entire control panel of the little Korg. If you've never played one, you might think that it is incredibly limiting; it offers no ADSR envelopes, no filter resonance controls, no obvious routing... in fact, little of anything. But appearances can be deceptive, and the 700 was responsible for many classic patches in the mid-1970s.

Setting it up to produce the desired waveform and filter roll-offs is trivial. First, we turn to the Scale and Mode selectors, setting the first to 4' (the violin is, after all, a fairly high-pitched instrument) and the latter to sawtooth. Next, we find the 'Traveler', which is a combined high-pass/low-pass filter. The upper slider controls a 12dB-per-octave low-pass filter, so we lower this somewhat to roll off the highest frequencies. Likewise, the lower slider controls a 12dB-per-octave high-pass filter, so we raise this to attenuate the low frequencies (see Figure 9 below).

If we now play the Korg through the RS360 filter bank, we obtain a sound that is nothing like a violin; in fact, it is closer to that of a banjo. This is because the envelope of the sound is percussive. We need to find the ADSR envelope generator and create something more in keeping with a 'bowed' sound.

Umm... except that the Korg 700 *has* no ADSR envelope generator. There are only three contour controls: the two sliders marked 'Attack/Slow' and 'Percussion/Singing', and the Sustain switch found in the group of eight switches near the centre of the panel. You'll note that each of these is coloured orange. This is not a coincidence. Korg colour-coded all the controls on their early synths: orange for amplitude controls, red for timbre controls, blue for pitch controls, yellow for the repeat LFO, and green for portamento.

We need to modify the contour of the sound so far obtained, and we do so by increasing the Attack to somewhere in the region of '5' or '6', and by increasing the Percussion to Singing, which is equivalent to a Sustain setting of 10 on a conventional synth. But beware... do not switch on Sustain by flipping the switch downward, because it applies a fixed amount of what we would normally term 'release' to the envelope, and that would be inappropriate for what we want (yes, I know that it's confusing, but that's part of the charm of the Korg 700; it does things differently, and forces you to approach sound creation in a novel manner). Anyway, we have now set the amplitude controls as shown in Figure 10 (on the next page), and obtained the amplitude contour shown in Figure 11. Things are starting to sound decidedly stringier.

So... how does it sound? Well, it has a 'bowing' sort of attack, and a similar timbre to a violin, but it still sounds little like a violin. It sounds like a synthesizer. On the other hand, if I were to pick up a violin and try to play it, the result would also sound nothing like a violin. The explanation for this is simple; I'm incapable of playing a bowed string instrument, and my attempts would surely incur the wrath of the Cats Protection League. Sure, I can scrape a basic tone from the instrument, but there is no finesse, no articulation, no feeling. And this is what is wrong with the patch I have just created. While a single note may sound

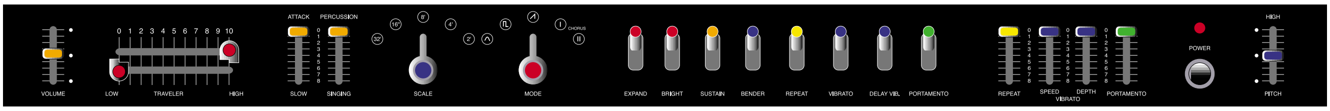


Figure 8: The starting patch for a Korg 700 minikorg.

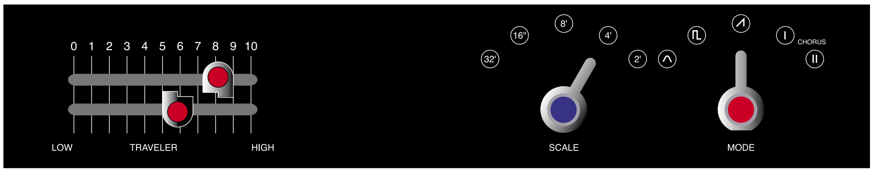


Figure 9: Setting the VCO and dual VCFs.

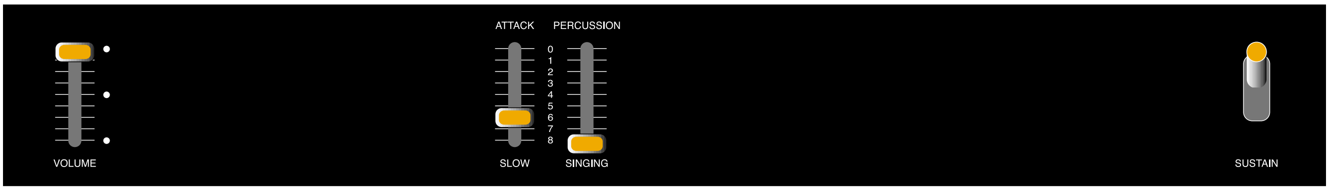


Figure 10: Setting the VCA.

vaguely like a bow being dragged across a violin string, a succession of notes have none of the attributes that make them sound like a musical instrument. So we're going to attempt to correct this by adding the most important element in the violinist's playing technique... vibrato.

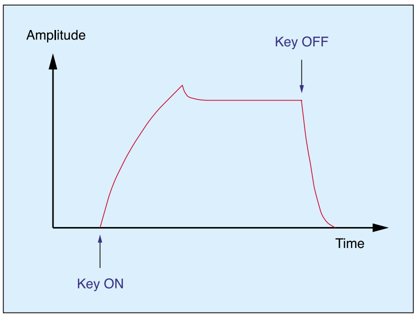


Figure 11: The unusual Korg 700 VCA contour.

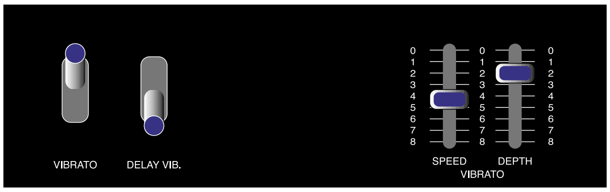


Figure 12: Adding delayed vibrato.

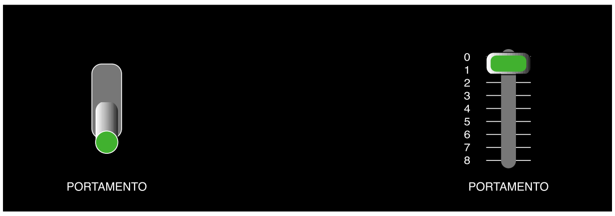


Figure 13: Adding a little glide to the patch.

A human violinist creates vibrato by wiggling the finger that is pressing the string against the neck of the instrument, The modulation speed is usually in the range of about 5Hz to 8Hz, and the amplitude can be surprisingly high; up to about a quarter of a tone. However, the vibrato is not consistent, and we should be aware that players tend to introduce it after the initial bowing action, and modify it to suit the requirements of the music.

Surprisingly, due to the physics of the violin, this vibrato also creates amplitude modulation (tremolo) and a good bowed string patch will take account of this. But the Korg 700 is not capable of tremolo, so we will have to ignore this.

Figure 12 (below) shows the vibrato controls for our patch. We choose modest settings for the vibrato speed and depth, adjusting these by ear to generate a pleasing amount of motion in the sound. Note that I have chosen to use the 'Delay Vib' setting in preference to straight 'Vibrato' because this introduces the effect a fraction of a second after you play the note. It's a crude imitation of a human performer, but it's far superior to having the vibrato present from the moment that you press a key.

The next problem concerns the pitching of the notes. This patch, like a piano or organ, plays notes discretely; that is, overlooking the vibrato for a moment, an 'A' is an 'A' from the moment that you press the key, a 'B' is precisely a 'B' and so on. This is not what happens when you play an unfretted string instrument, and even the best violinists will play notes a fraction sharp or flat, and then 'hunt' for the correct pitch. Add to that the glide used in violin performances, and it is clear that — to imitate the sound more accurately — we must introduce some sort of 'performance' attribute to the pitch.

On a more sophisticated synth, we could use the pitch-bend wheel to add glide, and even (with practice) to create a more musical vibrato. But the Korg 700 has no such controls; no pitch wheel, no joystick, no touchpad — no nuffink! What it does have, however, is portamento, and we can use the tiniest amount of this to create an almost imperceptible glide between notes (see Figure 13). Any more than the tiniest amount will destroy the illusion instantly, but if you can get the Portamento slider to sit just off 'zero', it can improve the patch considerably.

And that's all there is to it. So let's summarise: firstly, we used the synth to generate a sawtooth wave, and filtered it using the internal low-pass and high-pass filters to satisfy the requirements of Figure 5. Secondly, we chose contour settings that create a slight 'bowing' attack to the beginning of the sound, and which sustain it for as long as a key is depressed. Thirdly, in an attempt to reduce the rather synthetic nature of the sound, we added delayed vibrato, plus a smidgen of portamento to make the transitions between notes less like an organ and more like an unfretted instrument. Finally, and again in keeping with Figure 5, we passed the result through a three-band parametric 'formant' filter that provides a rough emulation of the most prominent body resonances of the real instrument.

So, does it now sound like a real violin? Don't be silly — it still sounds like a 1970s synth patch of a violin. Nevertheless, played sympathetically, and with careful tuning of the RS360, it has a pleasing 'violin-y' quality, particularly at higher pitches.

But what if you don't have a filter bank? Does the patch still work? Indeed it does, although it loses a little of the timbre that I've been trying to create. To demonstrate this, Figure 14 shows the original Korg 700 'factory' violin patch, and you can see both the similarities and differences to mine. For example, I prefer a slower Attack, and use the high-pass filter to remove more of the lower frequencies. What's more, I use less vibrato and have added that tiny amount of portamento, but the basis of the patch is essentially the same. Still, neither sound is the best imitation of a violin that I have heard produced by an analogue synth, so next month we'll investigate bowed string sounds further, and see what improvements we can obtain on more sophisticated equipment. SOS

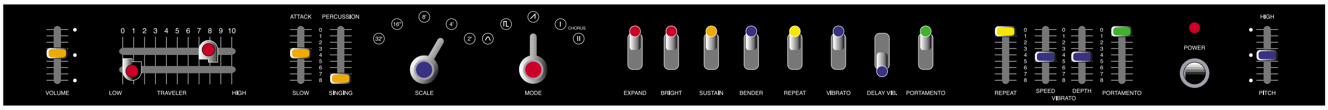


Figure 14: The 'factory' violin patch from the Korg 700 manual, printed in 1974.

# Synth Secrets

## Practical Bowed-string Synthesis Continued

Gordon Reid

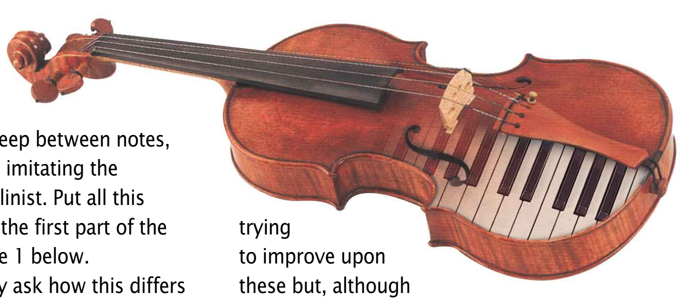
**After putting all our bowed-string synthesis theory into practice on a Korg 700 last month, we found that the result was only acceptable as a string sound with a *lot* of wishful thinking. Can we improve on it?**

Having investigated the audio properties of bowed instruments two months ago, we proceeded last month to craft a simple imitation of a violin on that most basic of analogue synthesizers, the Korg 700. If you tried this, you will have found that while being reminiscent of something vaguely stringy, the patch sounded nothing like a 300-year old instrument constructed from bits of dead trees and dead cats, all held together by bits of dead horses. This is not surprising... the Korg 700 was one of the earliest commercial synthesizers, designed from ideas developed in the late 1960s that were originally destined for inclusion in a combo organ.

Casting an eye over some of the other synths we've programmed over the past year or two, I can't see that the Roland SH101, ARP Axxe or Minimoog will offer us much more for this sound than the Korg 700. So, as I've done on a number of occasions in the past, I'm going to look to a rather more flexible analogue monosynth to see whether we can improve upon last month's result. This is the Korg MS20, whose twin filters, dual envelope generators and semi-modular modulation capabilities should offer possibilities unavailable on the other synths.

### Patching A Violin Sound On The Korg MS20

As always, we'll start with the oscillators. Deciding on the settings for these is simple; we know that a sawtooth wave is the initial waveform produced by all bowed instruments. Consequently, we choose the sawtooth setting for VCO1, set the pitch to 4' (remember, the violin is a relatively high-pitched instrument), and set the oscillator output to '10' in the Mixer. For reasons that are immediately obvious if we do anything else, we set the VCO2 output to zero, meaning that its waveform, pitch and footage settings are irrelevant. We will also add a little portamento, just as we did last month. You'll remember that this produces a quick, almost imperceptible sweep between notes, which goes some way to imitating the fingering of a human violinist. Put all this together and you obtain the first part of the patch, as shown in Figure 1 below.



At this point, you may ask how this differs from the Korg 700 patch. In truth, it doesn't. You can compare the settings on the two synths and, despite looking very different, they perform exactly the same functions. So let's move on to the filters. Surely, the MS20 will show its true strength here.

Strangely... it doesn't. If you experiment, you'll find that reducing the low-pass filter cutoff frequency reduces the brightness of the Korg's rather electronic-sounding oscillators (which is good) but that raising the high-pass filter until its effect becomes noticeable guts the sound, leaving you with a rather annoying buzz (which is bad). What's more, adding even the slightest amount of 'peak' (which is Korg's name for filter resonance, or 'emphasis') makes the patch sound electronic in nature. All of which means that my preferred filter settings are very simple, as shown in Figure 2 (right).

I spent a fair amount of time trying to improve upon these but, although theory suggests that we should raise the high-pass filter and add resonance to imitate the body resonances of the violin, the MS20's filters do not seem to be well suited to this. It proves — yet again — that not all synths are capable of all types of sound.

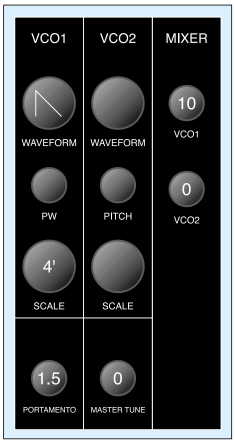


Figure 1: The oscillator settings for an MS20 violin patch.

Let's move on to the MS20's main envelope generator which, just to be arcane, is Envelope Generator 2. This produces a five-stage contour, of the HADSR type (Hold, Attack, Decay, Sustain, Release). Thinking about suitable values for each of these stages, we clearly don't want any Hold, because this is a delay before the Attack stage; once we press a key, we want the sound to begin. The rest of the envelope settings are less concrete. If you refer to the Korg patch library, the violin chart suggests that the values in Figure 3 (above right) are appropriate. Certainly, they are not inappropriate; the non-zero Attack and Release values ensure

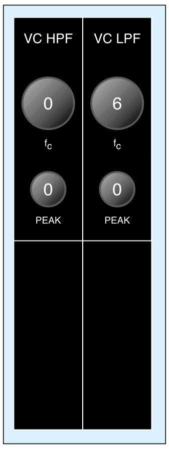


Figure 2: The filter settings for the violin patch.

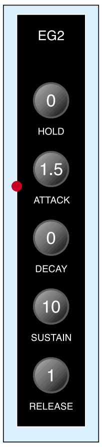


Figure 3: Korg's suggestions for the contour of a violin patch.

that the note does not have an organ-like character. However, I think that we can do a little better by creating a slight bump at the start of the sound, as determined by the settings in Figure 4 (below). You can see these two contours in Figure 5 and 6 (below right), and I'll leave it to you to decide which is the more suitable.

If you create the patch shown so far, but set everything else on the panel to zero, you'll obtain a vaguely stringy sound that lacks any form of life or interest. We encountered the same problem last month, and to some extent alleviated it by adding delayed vibrato. Unfortunately, unlike the simpler and (ultimately) less capable Korg 700, the MS20 has no delayed vibrato, nor does it seem to offer manual control over pitch or modulation depth. Sure, you can add vibrato using the Modulation Generator and the FM knobs in the main control panel, but the result is quite unnatural. So now we're going to use the MS20's patchbay to improve things.

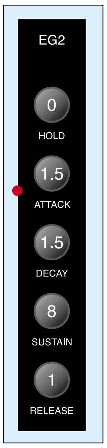


Figure 4: A revised contour for the violin patch.

## Semi-Modular Synthesis

Before we start shoving cables into inputs and outputs, let's remind ourselves about the difference between a semi-modular synth and its fully modular brethren. On the latter, the various sound-shaping modules are not connected to each other so, to obtain the sound you want, you must link things together in a way that directs the audio path through the modules that you want to use, and provides control signals as appropriate. This might sound like a lot of work, but a modular synth repays this by being far more flexible than a pre-patched synth with the same complement of modules. And, of course, you can also add bits and bobs as and when you need them (or can afford them), thus extending the capabilities of your synth as it grows. In contrast, semi-modular synths have a fixed number of modules that are pre-connected internally, thus allowing you to create sounds without patch cords. However, this class of instruments also allows you to alter the routing of some of its internal signals, connecting this output to that input, thus extending the range of sounds that you can obtain.

Some semi-modular synths are as powerful as their modular equivalents. For example, if you bought all the modules needed to recreate the architecture of the semi-modular ARP 2600, it's unlikely that you would improve much upon the original. On the other hand, the MS20 is far more limited because its audio signal path is predetermined, and you cannot interrupt it using cords. Despite the impressive appearance of the right half of the control panel, you can use this only to create new modulation routings; no matter what, the signal generated by the VCOs passes through the VCFs to the output VCA. Nevertheless, the MS20 offers everything that we need to create delayed vibrato.

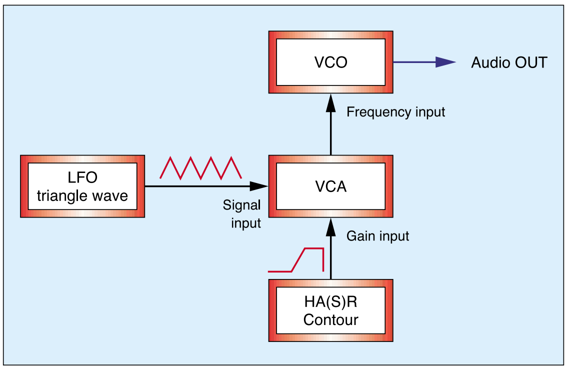


Figure 7: Creating delayed vibrato from discrete modules.

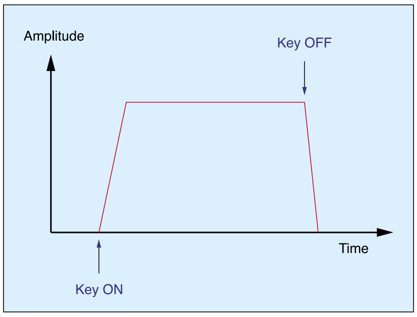


Figure 5: The contour generated by Korg's HADSR settings in Figure 3.

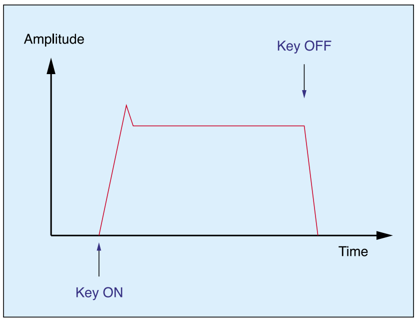


Figure 6: The contour generated by my HADSR settings in Figure 4.

Think about the fundamental nature of delayed vibrato. The modulation is in essence the same as normal vibrato, but it starts with zero amplitude and then, after a period, increases smoothly to some suitable depth. Figure 7 (above) shows the synth architecture needed to create this effect.

So now we'll configure the MS20 in this fashion. Referring to Figure 8 (below), look at the patch panel, and you'll see a box annotated 'MG' (Modulation Generator). This offers two outputs, one for saw and triangle waves, the other for pulse waves. We can direct a triangle wave of suitable frequency (as determined by the knobs in the MG section) to the input of the patchable VCA in the patch panel. This is the lower of the red

▶

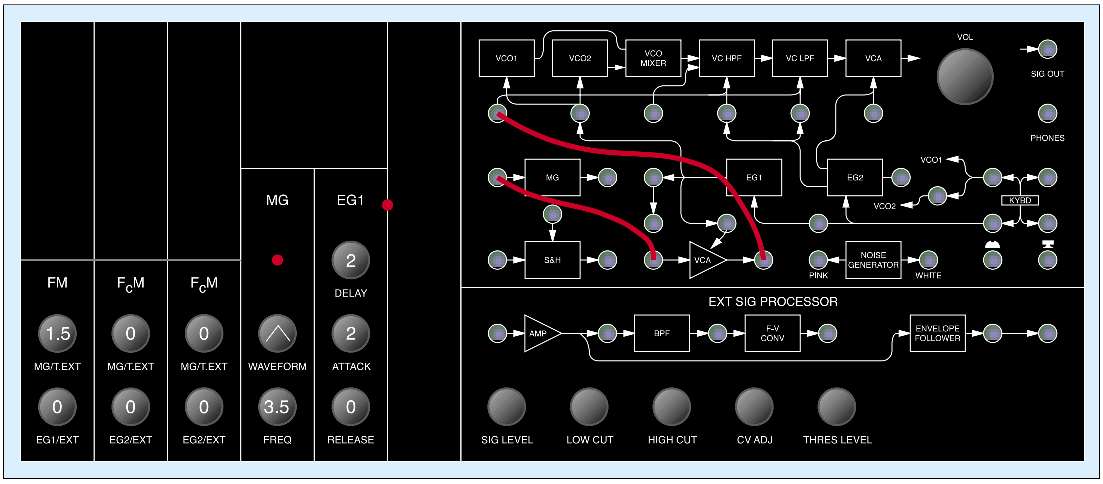


Figure 8: Patching delayed vibrato.

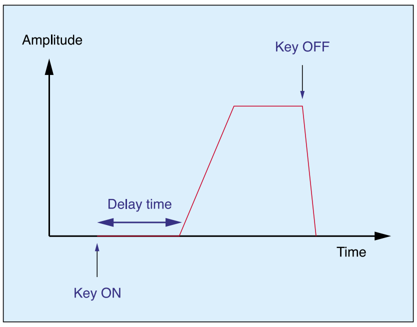


Figure 9: The HA(S)R contour generator.

patch cables in the diagram.

Next, we need to determine what the LFO is controlling, and we do so by taking the output from the VCA to the 'Total' input at the top left of the patch panel. This is the upper of the red cables shown in Figure 8. Any signal presented to the Total input will modulate the pitches of VCO1 and VCO2, as well as the cutoff frequencies of the low- and high-pass filters, with the modulation amplitudes determined by the positions of the top row of knobs in the modulation sections. Given that we have no wish to create a wow effect, we set the first of the three 'MG/T.Ext' knobs for gentle frequency modulation (I have used a value of 1.5) but the second and third to zero so that there is no filter modulation.

Hang on... what does 'MG/T.Ext' mean? If no cable is inserted into the Total input, these knobs control the amount of Modulation Generator signal applied directly to the oscillators and each of the filters. But if a cable *is* inserted, the internal patching is broken, and the signal carried by the cable — which, in this case, is the modified output from the Modulation Generator — is used. This breaking of the internal patching and the inserting of new CVs is the very essence of programming semi-modular synths.

The modulating waveform is now passing through the VCA and onward to control the pitch, so we now need to decide what we're going to use to control the amplitude of the modulation signal.

Happily, the MS20 offers the perfect CV generator for delayed vibrato: the HA(S)R contour generator called Envelope Generator 1. This has three controls; the Hold time, Attack rate and Release rate. I have added the (S) in my description to make it clear that the contour remains at its peak level once the Attack is complete and until you release the key (see Figure 9, left).

All we have to do, therefore, is patch the output from EG1 to the input of the VCA. Except that we don't... As the graphics on the MS20's panel show, this has been done internally for us.

Before moving on from the modulation panel and controls, take a peek at the 'EG2/Ext' knob in the low-pass filter panel. This currently has a value of zero. Indeed, all four filter modulation knobs are at zero, so nothing on the synth is controlling the filters' cutoff frequencies. This agrees with what we know about the violin but, in a departure from the theory, Figure 10 (above) shows the patch panel with a third (yellow) cord inserted. This leads from the Keyboard CV output to the 'LPF Cutoff Freq' input, meaning that cutoff frequency will track as you play up and down the keyboard, becoming brighter at high pitches, and duller at low pitches. The amount by which this happens is determined by the aforementioned 'EG2/Ext' knob, and I find

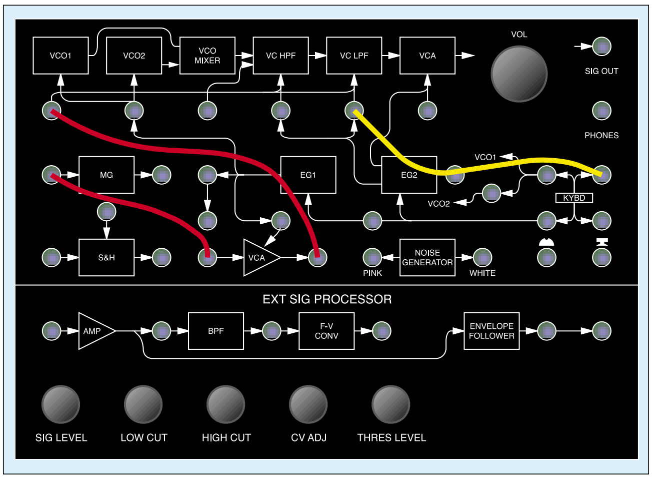


Figure 10: Adding keyboard/filter tracking.

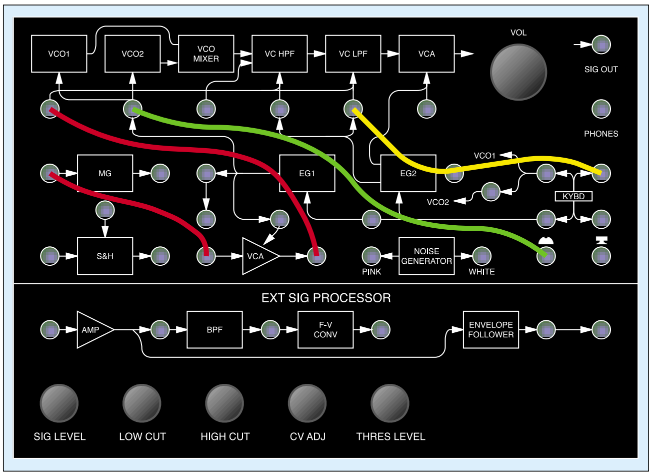
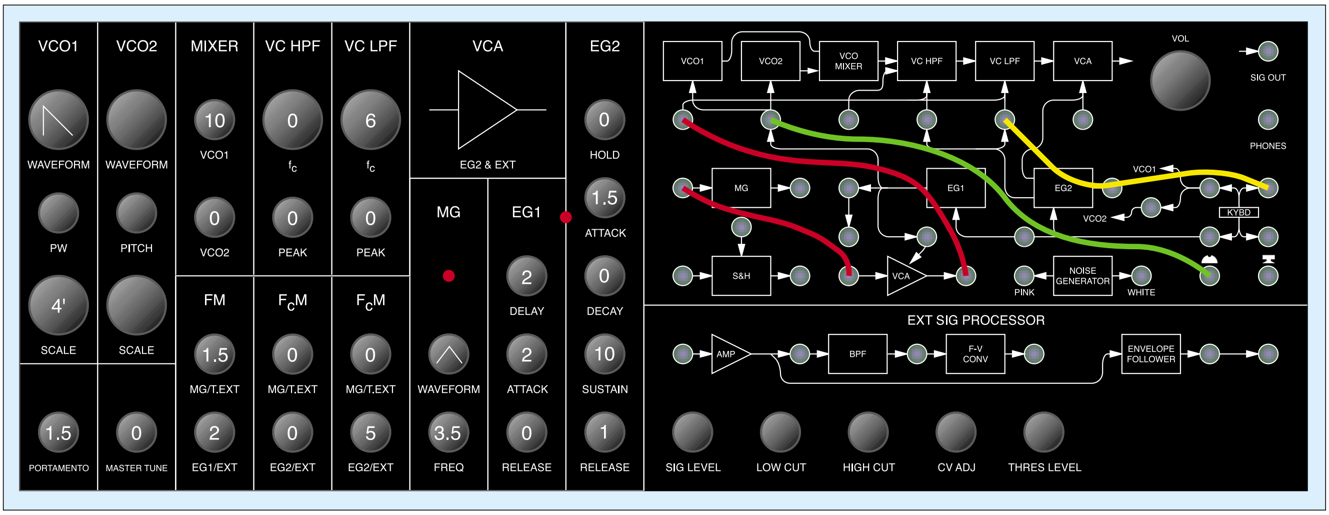


Figure 11: Adding pitch-bend using the wheel.

Figure 12: An MS20 violin patch.



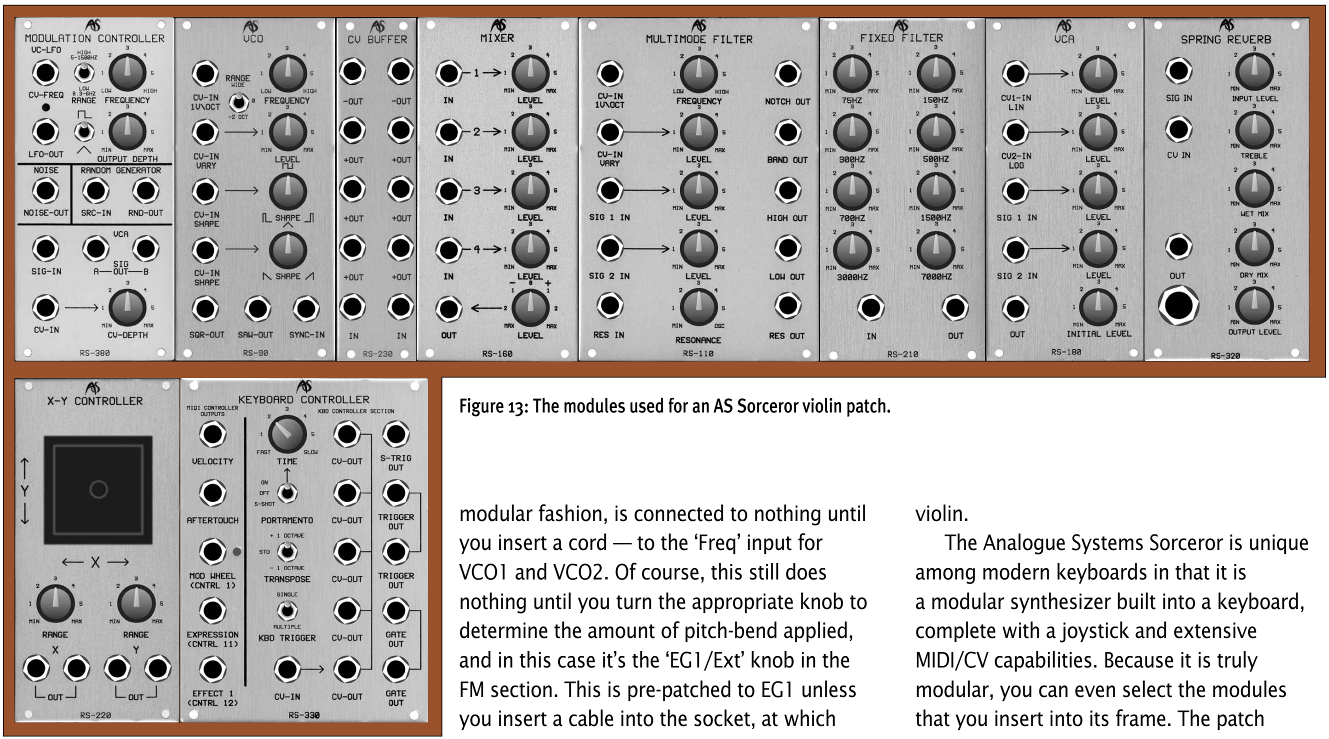


Figure 13: The modules used for an AS Sorceror violin patch.

a value of '5' to be most agreeable.

So, what's left to do? There is just one, simple modification that I think improves the playability of this patch. Although we have taken modulation duties out of human hands and handed them over to the combination of an MG and VCA, the MS20 has no pre-patched pitch-bend capability. We have to add this using yet another cable, which I have shown in green in Figure 11 (on the previous page). This links the control wheel — which, in true modular fashion, is connected to nothing until you insert a cord — to the 'Freq' input for VCO1 and VCO2. Of course, this still does nothing until you turn the appropriate knob to determine the amount of pitch-bend applied, and in this case it's the 'EG1/Ext' knob in the FM section. This is pre-patched to EG1 unless you insert a cable into the socket, at which point the internal connection is broken, and the signal carried by the patch cord is passed. I find that a value of '2' is most useful since, by coincidence, this attenuates the wheel's signal so that the pitch-bend is ±2 semitones, which is musically pleasing.

So there we have it... and Figure 12 (on the previous page) shows the complete MS20 violin patch in all its glory. Played sympathetically, it can be slightly evocative of the original instrument and, although nobody is going to mistake it for Stephane Grappelli or Yehudi Menuhin, it works reasonably well in slow, solo passages.

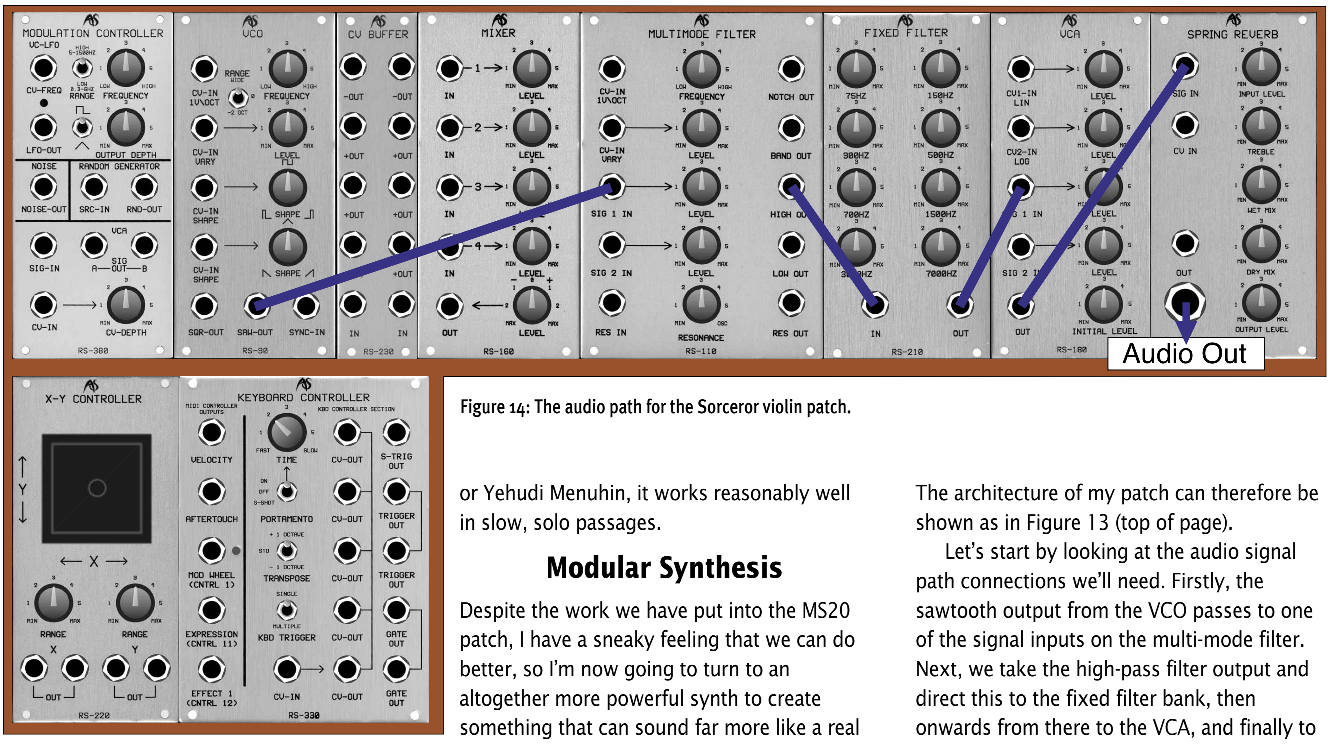


Figure 14: The audio path for the Sorceror violin patch.

## Modular Synthesis

Despite the work we have put into the MS20 patch, I have a sneaky feeling that we can do better, so I'm now going to turn to an altogether more powerful synth to create something that can sound far more like a real violin.

The Analogue Systems Sorceror is unique among modern keyboards in that it is a modular synthesizer built into a keyboard, complete with a joystick and extensive MIDI/CV capabilities. Because it is truly modular, you can even select the modules that you insert into its frame. The patch I intend to construct with it this month requires the following modules:

- RS220 X-Y (joystick) controller.
- RS330 keyboard controller.
- RS380 modulation controller.
- RS90 VCO.
- RS230 CV buffer.
- RS160 mixer.
- RS110 multi-mode filter.
- RS210 fixed filter.
- RS180 VCA.
- RS320 spring reverb.

The architecture of my patch can therefore be shown as in Figure 13 (top of page).

Let's start by looking at the audio signal path connections we'll need. Firstly, the sawtooth output from the VCO passes to one of the signal inputs on the multi-mode filter. Next, we take the high-pass filter output and direct this to the fixed filter bank, then onwards from there to the VCA, and finally to the spring reverb before the audio reaches the outside world. I've shown these connections as blue cords in Figure 14.

You may wonder why — in addition to the fixed filter bank — I have used the high-pass option in the multi-mode filter. The answer lies in Figure 15 (right), which I've reproduced from my original analysis of the violin's frequency response. This shows a couple of predominant low-frequency resonances, a plateau across a few thousand Hertz, and a gentle roll-off at high frequencies.

As discussed last month, a Formant Filter is perhaps the best tool for recreating this, but we can generate the general shape quite efficiently using a fixed filter bank to create the plateau and the low- and high- frequency roll-offs, while the high-pass filter — with its resonance set just a tad below self-oscillation — creates a single, large, low-frequency resonance. Using the trick I described last month, we can set the filter's cutoff frequency to about one-and-a-half octaves above middle 'C', which will lie in the region of 700Hz. Then, we need only adjust the eight filters in the RS210 to obtain the response shown in Figure 16 (right). Note that, unlike the parallel band-pass filters that I used last month, the high-pass filter and fixed filter bank in this patch are in series. This means that we do not add the responses together, we multiply them to obtain the combined response.

Now let's look at the control CVs, as shown in Figure 17 below. Unlike the audio path connections, these look like the beginnings of a bird's nest, and will require some explanation.

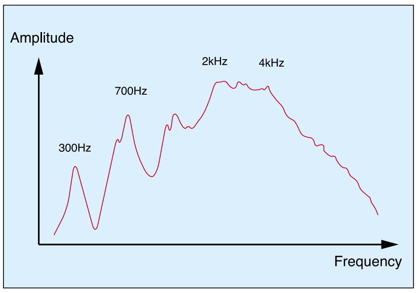


Figure 15: The frequency response obtained when energising the violin body using a string, bow, and bridge.

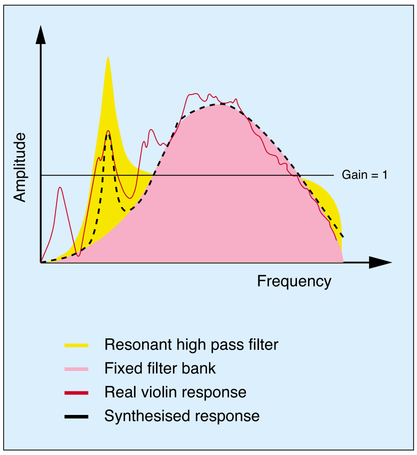


Figure 16: Using a fixed filter and a resonant high-pass filter to imitate the violin's frequency response.

Let's start with the red patch cords. These show that the output from the LFO in the Modulation Controller is directed to the patchable VCA within the same module. The output from this passes to the Mixer, and then to one of the pitch CV inputs of the VCO.

Clearly, the LFO is generating vibrato, so why does it pass along such a convoluted path? To answer this, consider the green cords. These show the path of the CV generated by the 'Y' direction of the joystick. This voltage passes first to the CV input of the CV buffer, whereupon it is split and sent to two destinations: the CV-IN of the VCA controlling the LFO depth, and the CV-IN of the VCA.

Looking at the first of these routings, we can see that the vibrato depth is controlled by the 'Y' direction of the joystick, just as it would be if we used a modulation wheel on a conventional synth. That's simple, and neat. But the second path is more surprising... it shows that the level of the audio signal (ie. what we're hearing) is also controlled by the 'Y' direction of the joystick. Unlike other synth patches we've discussed in Synth Secrets, there's no contour generator shaping the sound. If you want to hear something, you must press a key *and* move the joystick!

You should not underestimate the importance of this. For the first time, I've placed real control of the sound in your hands. If you use the joystick well, you can articulate the notes with feeling and real sympathy for the instrument you're imitating. If you use it badly, the patch will sound... well, bad.

There are two other cords in Figure 17. The yellow one takes the keyboard CV and passes it to the standard 1V-per-octave CV input of the oscillator so that we can play the keyboard and obtain the expected pitches from our 'violin'. The purple one takes the CV generated by the 'X' direction of the joystick and mixes this in the Mixer with the modulation signal from the RS380. This means that you use the joystick to bend notes as well as articulate them and add vibrato.

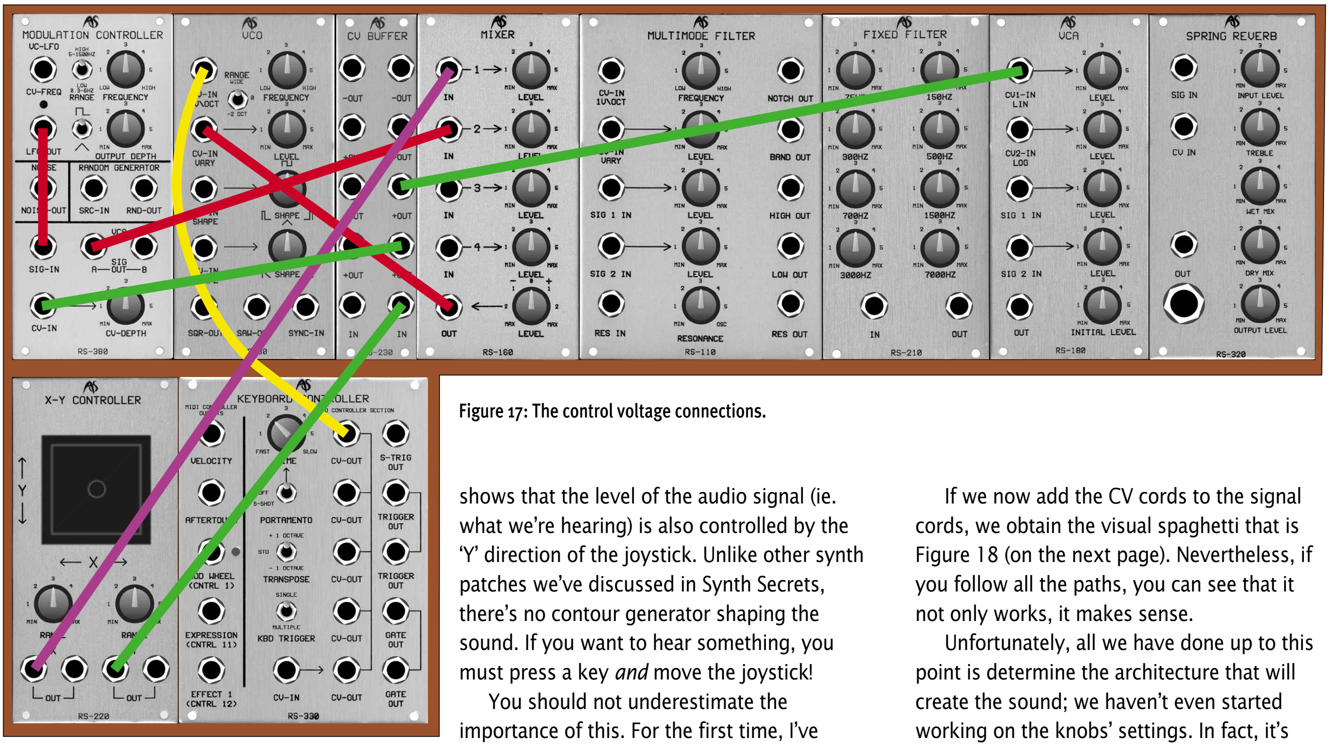


Figure 17: The control voltage connections.

If we now add the CV cords to the signal cords, we obtain the visual spaghetti that is Figure 18 (on the next page). Nevertheless, if you follow all the paths, you can see that it not only works, it makes sense.

Unfortunately, all we have done up to this point is determine the architecture that will create the sound; we haven't even started working on the knobs' settings. In fact, it's

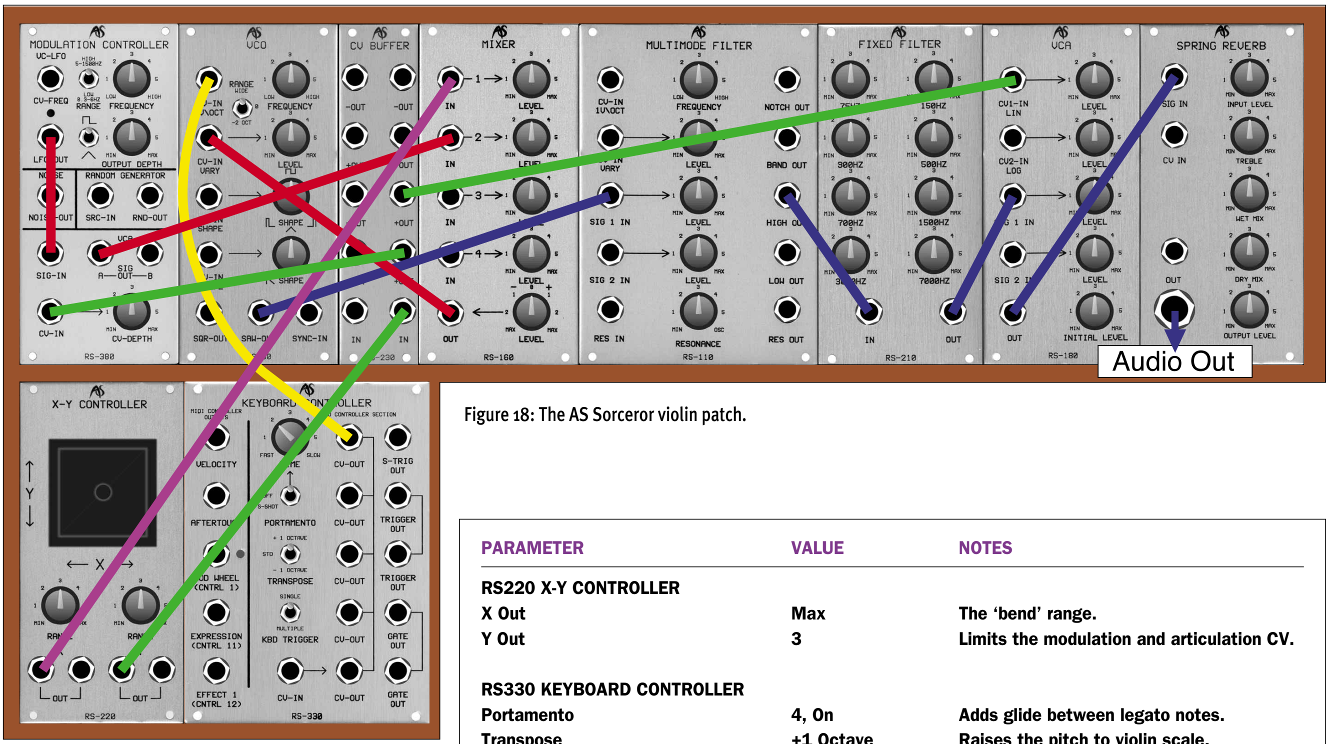


Figure 18: The AS Sorceror violin patch.

▶ a bit like taking a Minimoog and expecting it to sound like a violin, simply because its modules are connected together. So let's finish this month's article by adding some values to the knobs in Figure 18. Bear in mind that these are to my taste on the days that I wrote this, and that you might find other values more to your liking. What's more, tiny changes in the values of the knobs can make significant differences to the resulting sound, yet I have listed them to the nearest half. So, with these caveats in mind, take a look at the table of values on the right. To obtain the complete patch, combine the routings depicted in Figure 18 with the settings in the table.

Press a key… and shock! horror!! Nothing comes out. Of course not, silly. I forgot to move the joystick forwards. Doing so introduces the note slowly, and adds vibrato progressively as it does so. Hey… this is good stuff! In fact, it's the closest that we have yet come to recreating the sound and feeling of a real violin. The reason for this is simple… using the joystick in this manner allows us to articulate notes in a far more 'human' fashion than is possible using envelopes. We can introduce notes slowly or quickly as desired, and with pitch-bend. Furthermore, we can imitate the bowing speed and pressure during the course of the note and, with careful use of the joystick, even add other effects as the note progresses. All in all, this proves that it's not the complexity of the patch that creates the performance. It's… well, the performance. And while that may seem obvious, it's surprising how often it seems to be forgotten! SOS

| PARAMETER | VALUE | NOTES |
|---|---|---|
| **RS220 X-Y CONTROLLER** | | |
| X Out | Max | The 'bend' range. |
| Y Out | 3 | Limits the modulation and articulation CV. |
| **RS330 KEYBOARD CONTROLLER** | | |
| Portamento | 4, On | Adds glide between legato notes. |
| Transpose | +1 Octave | Raises the pitch to violin scale. |
| **RS380 MODULATION CONTROLLER** | | |
| LFO Waveform | Triangle | |
| LFO Rate | Low, approx 4.5Hz | A natural vibrato speed. |
| LFO Depth | 3 | Controls the initial vibrato depth. |
| CV Depth | 4 | Controls the amount of modulation. |
| **RS90 VCO** | | |
| Frequency | 2.5 | |
| CV-In Vary Level | 0.5 | Ensures that the vibrato is gentle. |
| Shape | Sawtooth | The correct waveform for bowed strings. |
| **RS160 MIXER** | | |
| Sig1 In Level | Max | Controls the amount of 'bend'. |
| Sig2 In Level | 1 | Controls the amount of modulation. |
| Out Level | +1.5 | |
| **RS110 MULTI-MODE FILTER** | | |
| Frequency | 3 | Approximately 700Hz. |
| Sig1 In Level | 3 | |
| Resonance | 5 | Just on the limit of 'ringing'. |
| **RS210 FIXED FILTER** | | |
| 75Hz | Min | |
| 150Hz | 3 | |
| 300Hz | 3 | |
| 500Hz | 4 | |
| 700Hz | 4.5 | |
| 1,5kHz | Max | |
| 3kHz | Max | |
| 7kHz | Min | |
| **RS180 VCA** | | |
| CV1 In Level | Max | |
| Sig1 In Level | 3.5 | |
| Initial Level | Min | No signal passes without 'Y' movement. |
| **RS320 SPRING REVERB** | | |
| Sig In Level | Max | |
| Treble | Max | |
| Wet Mix | 1 | Just a smidgen to soften the sound. |
| Dry Mix | Max | |
| Output Level | ? | Whatever suits your amplifier. |

# Synth Secrets

## Articulation & Bowed-string Synthesis

Gordon Reid

**The skilful articulation of a synthesized string patch can improve it no end, even one created using very basic building blocks, as we saw at the end of last month. But we can take this approach much further...**

Nowadays, when people talk about synthesizers and synthesis, they invariably talk about keyboard instruments. They may ask each other what type of synths they're using, but this will usually be a question about the nature of the sound generator... is it analogue, PCM- or sample-based, FM, additive, 'virtual' analogue, physically modelled, or a host of lesser variants. Alternatively, the question may refer to whether the instrument is monophonic or polyphonic, or it may be an enquiry about the manufacturer or model. But, with the exception of a handful of modules designed for use with wind controllers or guitar pitch-to-MIDI converters, synthesizers are almost always keyboards or modules controlled via CV and Gate signals or MIDI from a keyboard. Consequently, the vast majority of synths play discrete notes, with exactly 12 of them per octave, each tuned in accordance with the strict set of frequency relationships that comprise 'equal temperament'. Of course, you can often control electronic instruments using computers and/or sequencers, and some newer models offer control over aspects of the sound using innovations such as Roland's D-Beam, but the musical philosophy and the temperament of the notes remain the same.

### The Tyranny Of Discrete Notes

It wasn't always like this. You only have to travel back 30 years or so to find bands such as Roxy Music and Hawkwind using synths to create new sounds that were part of the music, but were in no way played chromatically. You could argue that this was making a virtue of necessity — the EMS VCS3 that both bands used was the least likely to stay in tune of all the early synths — but this would be a simplistic argument. Brian Eno and Del Detmar could just as easily have bought a Moog or ARP and played widdly melodies as most of their contemporaries were doing. But, for a couple of years, a-melodic synthesis was not only acceptable, it was an exciting area of musical exploration.

Unfortunately, far from being the dawn of a new age of music, the 1970s proved to be the end of this fascinating era of synthesis. Despite numerous turns in the cycle of musical fashion, from electro-pop to industrial electronica, from New Age to dance, the music of the past 30 years has been firmly rooted in the 12-note chromatic, equal-tempered scale fully explored by Johann Sebastian Bach in the 18th century. Sure, *Sound On Sound*'s readers generally make more use of rhythm instruments than did the Baroque composers of Bach's era, and — for the most part — we make less use of melody and counterpoint, but 200 year-old central European music scholars would probably find no difficulty comprehending the form of today's popular music. Like my mother, they would undoubtedly hate it — but they would understand it.

This is stranger than it might seem. Whereas pianos and organs can only play discrete notes locked to a chosen scale, many traditional instruments are less constrained. All non-fretted string instruments allow you to play any pitch within their ranges, and many brass and woodwind instruments allow you to slide between the semitones defined by their holes or valves. And then there's the trombone, but I'm not sure that we should mention this in polite company.

So how did Western music become so firmly locked into forms of music limited by notes of discrete durations and specific pitches, all of which conform to the well-tempered scale? It didn't happen in India, nor in Bali... countries whose wonderful music sounds so 'wrong' to most Western ears. In all likelihood, it's the result of exposure to one musical form from birth, resulting in our stunted appreciation of what constitutes a note, what constitutes a musical interval, and what frequencies are acceptable in a melody. All of which brings us back to the synthesizer — an instrument that is in principle so flexible that it can emulate all the musical forms known, and be the inspiration for quite a few new ones, were it not for the limited imaginations of those who use them, myself included.

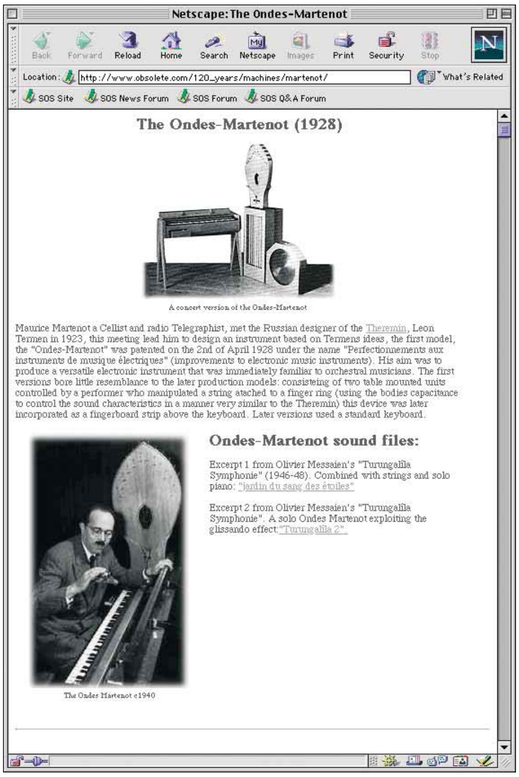
Now, you could point out that many modern synths offer alternative temperaments such as Pure Major, Pure Minor, Just, and Werckmeister III, but these are different ways of tuning the 12 keys that comprise each octave. The underlying philosophy remains unchanged.

To find an instrument that breaks this mould, you might turn (as did I) to the experimental synthesizers developed during the first six decades of the 20th century. Some of these eschewed keyboards in favour of other mechanical systems for determining pitch, tone, duration and loudness but, if you inspect them closely, you'll find that many used paper tape with punched holes to play notes according to... conventional scales and temperament. In other words, a set of electronic sensors replaced the musician's fingers, but the musical philosophy was again unchanged.

Perhaps the only well known exceptions to the tyranny of the 12-note octave were the instruments made by Don Buchla in the 1960s and 1970s. His 'System 100' incorporated a sub-divided touch pad that made no concessions to the standard keyboard geometry, and allowed you to tune each division independently. Nevertheless, this *still* forced the player to think in terms of discrete notes with fixed divisions of pitch, with pitch-bend or slew to generate frequencies that lay between the notes.

As far as I am aware, there were — and remain — only two electronic instruments that challenge the domination of sub-divided pitch and discrete articulation of the notes. Both were developed in the early part of the last century, and both have found favour within all musical genres including classical music, experimental, jazz and *avant-garde* music, plus pop and rock. They are the Theremin and the Ondes Martenot.

Of the two, the Theremin is by far the better known, and over the past few years there have been many models produced. These range from small, basic, single-antenna boxes that cost a few pounds, to the popular 'Etherwave', through to Bob Moog's expensive recreations of the original, floor-standing instruments.



Netscape: The Ondes-Martenot

Location: http://www.obsolete.com/120_years/machines/martenot/

The Ondes-Martenot (1928)

A concert version of the Ondes-Martenot

Maurice Martenot a Cellist and radio Telegraphist, met the Russian designer of the Theremin, Leon Termen in 1923, this meeting lead him to design an instrument based on Termens ideas, the first model, the "Ondes-Martenot" was patented on the 2nd of April 1928 under the name "Perfectionnements aux instruments de musique électriques" (improvements to electronic music instruments). His aim was to produce a versatile electronic instrument that was immediately familiar to orchestral musicians. The first versions bore little resemblance to the later production models: consisteing of two table mounted units controlled by a performer who manipulated a string atached to a finger ring (using the bodies capacitance to control the sound characteristics in a manner very similar to the Theremin) this device was later incorporated as a fingerboard strip above the keyboard. Later versions used a standard keyboard.

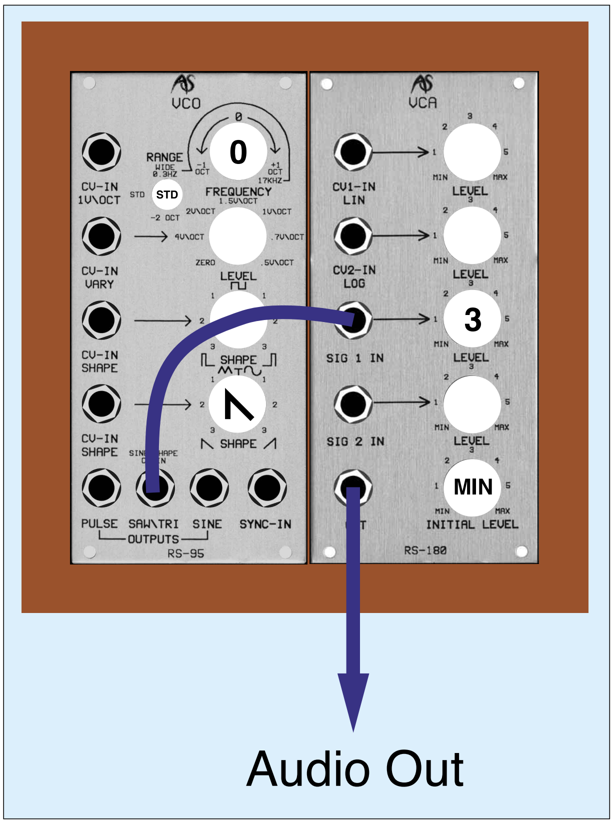
Ondes-Martenot sound files:

Excerpt 1 from Olivier Messaien's "Turungalila Symphonie" (1946-48). Combined with strings and solo piano: "jardin du sang des étoiles"

Excerpt 2 from Olivier Messaien's "Turungalila Symphonie". A solo Ondes Martenot exploiting the glissando effect:"Turungalila 2".

The Ondes Martenot c1940

Figure 1: A very simple patch.



Nevertheless, we're not going to discuss the Theremin this month. Instead, we're going to concentrate on the Ondes Martenot, and show how an obscure musical controller invented three-quarters of a century ago might improve many aspects of your synthesis technique in 2003, particularly with regard to our bowed-string patches.

## Another Way To Play Synths

The Ondes Martenot, unveiled by Maurice Martenot in 1928, is a fascinating instrument, and if you're not familiar with it, it's really worth investigating. One of the best Internet-based resources for obscure 20th-century electronic instruments is www.obsolete.com/120_years/, and you'll find a page devoted to this unique device at www.obsolete.com/120_years/machines/martenot/ (see left). Those unfamiliar with it might well ask what is so different about it — it is, after all, another keyboard-based instrument. But the Martenot has *two* pitch-control mechanisms. The first of these is its conventional keyboard, but we'll ignore this from now on. The second is a little ring attached to a wire that runs along the front of the instrument. This makes the Ondes Martenot almost unique among electronic instruments because, if you slip your finger through the ring and move it to the right, the pitch of the instrument rises without quantised steps; if you move it to the left, the pitch falls without quantised steps. The keyboard then becomes no more than a reference, letting you see where you are *vis-à-vis* the conventional scale, but you are no longer constrained by its discrete divisions: you can move the ring to any position you choose. This, of course, is the electronic equivalent of a fretless stringed instrument.

OK... so the wire and ring let you control the pitch, but how do you get a sound out of an Ondes Martenot? Or, more pertinently, having done so, how do you shut the thing up? The secret lies in a second control found to the left of the keyboard/ring/wire assembly; a large, wooden button that controls the loudness of the sound. When this is 'out' (ie. when you're not pressing it) the Ondes Martenot produces no sound. As you press the button in, the sound becomes louder until, when the button is fully depressed, the sound is at its loudest.

With careful use of the two controls, you can pitch and articulate notes in ways that are impossible on a conventional synth. Sure, we got close last month by using a joystick to control the loudness, but it's not quite the same, believe me.

All of which brings us neatly back to the string sounds we were discussing in the last couple of instalments of Synth Secrets, and the point at which I left you last month.

## More On String Synthesis

Let's start by considering the patch in Figure 1 above. This must rank among the simplest of all possible patches, with two modules — in this case, a voltage-controlled oscillator and a voltage-controlled amplifier — connected by a single cord. At this point, there's no keyboard attached, so there seems to be no way to determine the pitch of the note, nor to determine its start and end times. However, if you turn the amplifier's Initial Level knob away from its ▶

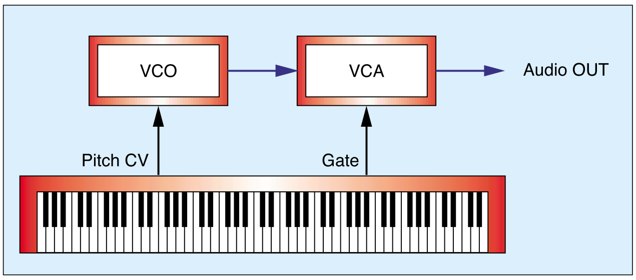


Figure 2: Connecting a keyboard to the modules in Figure 1.

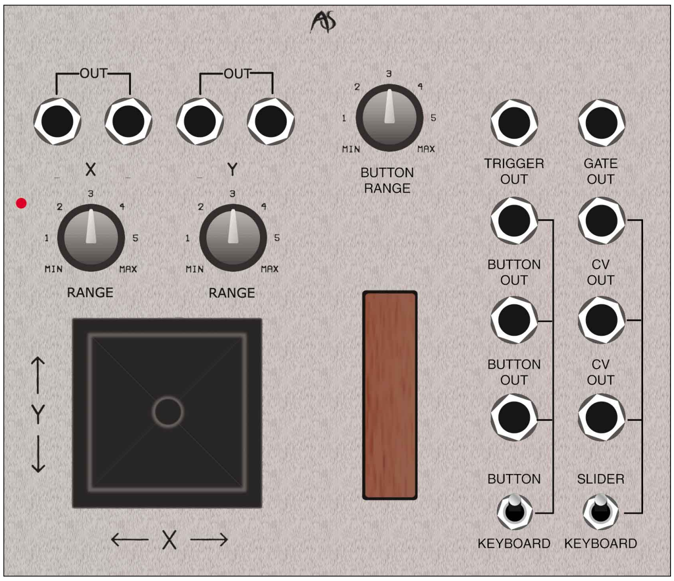


Figure 5: The control panel of the French Connection.

minimum, you will obtain a note whose tone is determined by the oscillator's Shape knob, and whose pitch is determined by the Frequency knob above it.

On the most basic level, these three knobs provide everything you need to create musical performances. If you could twist them quickly enough and accurately enough, you could make precise changes in pitch and tone, and articulate the resulting sounds as you pleased, just as if you were using envelope generators, pitch controllers, modulation generators, and all the other bits and pieces that comprise an integrated analogue synthesizer.

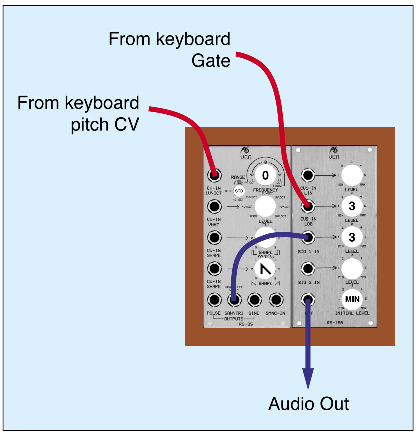


Figure 3: Adding patch cables to control the pitch of the note and when it sounds.

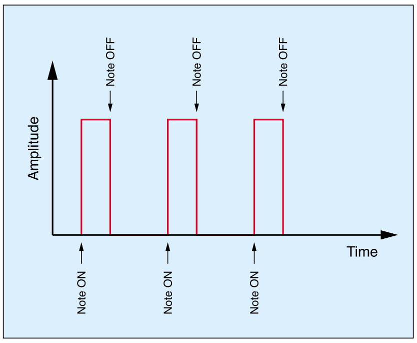


Figure 4(a): Using the Gate itself to control the gain of a VCA, resulting in 'organ-like' notes.

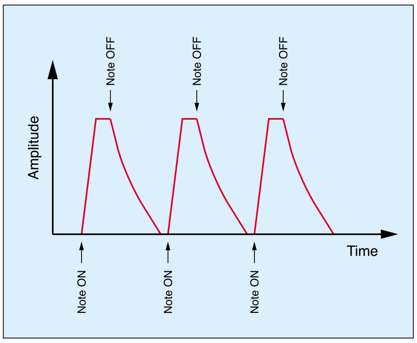


Figure 4(b): Using a contour generator to control the gain of a VCA.

Unfortunately, this is not simple. In fact, it's all but impossible, so we add a controlling mechanism to do most of the work for us. And, as discussed above, this mechanism is almost invariably a keyboard; perhaps with modulation and pitch-bend wheels that help us to inject some humanity into the performance, but a keyboard nonetheless. So we end up with the architecture shown in Figure 2 (top left), and the connections shown in Figure 3 (left). Note that I have connected the keyboard Gate directly to control the amplifier. This is an acceptable practice because synthesizers with +5V Gates usually have +5V or +10V CV inputs on their VCAs, so you will do no damage. Sure, you lose the shaping capabilities you obtain when you use the Gate to trigger a contour generator and then use the resulting envelope to control the VCA, but the square 'organ-like' notes produced by the Gate itself are acceptable because they are, well... like playing an organ (see Figures 4(a) and 4(b) left).

If we play the patch in Figures 2 and 3, we obtain notes with sawtooth waveforms and the predicted organ-like articulation. It's perfectly useable, and no doubt would have graced many 1970s prog-rock recordings, but despite being based on the appropriate waveform, it sounds nothing like a bowed, stringed instrument.

So let's replace the keyboard with a version of the Ondes Martenot that has been developed specifically for use as a controller of analogue synths. There is only one such beastie in production — the French Connection from Analogue Systems, shown below — and this offers the ring/wire controller and the amplitude button of the original instrument, as well as an X/Y joystick controller (for more on the French Connection, as well as much more on the Ondes Martenot on which it is based, take a look at the review back in *SOS* February 2002, or head for www.sospubs.co.uk/sos/feb02/articles/frenchconnection.asp).

Figure 5 (above) shows the French Connection's control panel. As you can see, there are two switches to the lower right of this, and if we were to flip both of these to 'Keyboard' and connect CV and Gate cables to the appropriate sockets, there's nothing stopping us from using the instrument as a conventional CV/Gate keyboard. I've shown the resulting patch in Figure 6, and

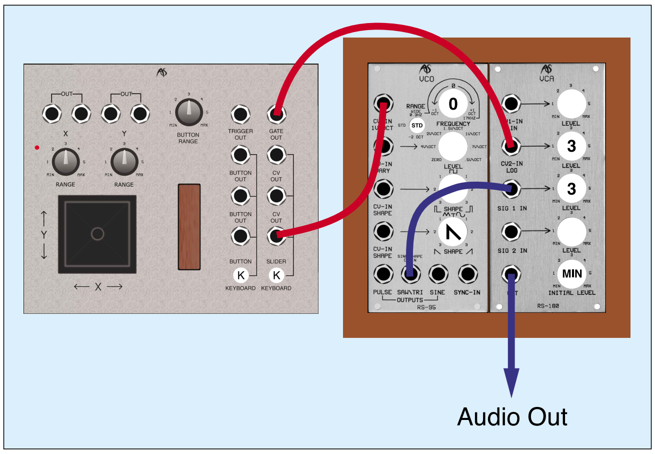


Figure 6: Controlling the notes using the French Connection's keyboard.

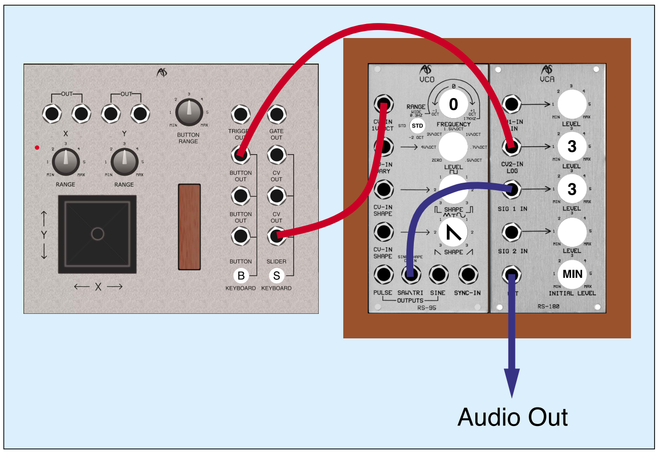


Figure 7: Controlling the patch using the French Connection's wire and button.

you can see that this is identical to that shown in Figure 3, so it will come as no surprise to you that it produces the organ-like sound already discussed.

However, if we now flip the switches to Button and Slider, playing the keyboard produces no sound whatsoever (see Figure 7 above). We must re-patch the cord in the Gate Out and insert it into one of the Button Outs. If we then press the button, we produce a variable voltage that controls the VCA Gain, and therefore the loudness of the sound. Likewise, playing the keyboard no longer controls the pitch. That duty is now undertaken by the wire, with the position of the ring determining the pitch at any given moment.

If we now play, the result is remarkable. With a little practice, the performance is no longer that of a soulless single-oscillator, unmodulated sawtooth buzz. You can add vibrato by wiggling your 'ring' finger from side to side, controlling both the speed and depth in a way that feels and sounds completely natural. Glide is merely a matter of pressing the button as you move to the next note. Moreover, judicious use of variable pressure on the button allows you to articulate notes in human ways, making each note unique in a fashion that is not possible when triggering envelope generators.

So, how does it sound? As you probably have guessed, the unquantised nature of the pitch controller, and the fluid way in which you can articulate sounds, means that the Ondes Martenot lends itself to imitations of unfretted string instruments. With just an oscillator and an amplifier, higher pitches sound remarkably like a violin, while lower pitches sound much like a contrabass or cello, although probably not one that the late, great Jacqueline du Pré would have cherished unduly.

The modern French Connection lacks the amazing resonant sound reproduction system of the original Ondes Martenot, but it nevertheless allows you to articulate and add expression to the almost limitless range of sounds available from any synthesizer that provides a pitch CV input and a VCA Gain input. Unfortunately, this precludes most common analogue (and digital) instruments, but if you own a modular or semi-modular synth, you're in business.

Of course, we need not stop here, and there's nothing preventing us from using the 'feel' of the French Connection with more complex patches. Figure 8 (below) shows a small extension to Figure 7 in which we control the waveform using the 'X' direction of the joystick. We achieve this by connecting a cable from one of the 'X' outputs to the sawtooth CV-In Shape input on the oscillator, so that left-to-right movements of the joystick change the wave from a sawtooth, to a triangle, to a ramp wave, and back again.

Playing this patch is surprisingly easy... use the inside of your left index finger to move the joystick while you use your thumb to press the button. Now you can articulate the note *and* determine its harmonic content with one hand, while playing the pitch with the other. It sounds simple, and it is, but this is something that you will find almost impossible on most synths. What's more, it's hugely expressive, because you can reduce the amplitude or even eliminate harmonics by moving the wave from a sawtooth towards a triangle as you reduce the overall loudness of the sound. This relationship between loudness and high-frequency content is — as we have discussed before in Synth Secrets — very much the behaviour of blown, bowed, strummed and struck instruments, and we're recreating it without a filter anywhere to be seen. Neat, huh?

But hang on a moment... Haven't I spent the last couple of months telling you that you need low-pass and high-pass filters, formant filters, modulation generators, mixers, joysticks, reverb, and loads of other gubbins to create even the barest likeness of

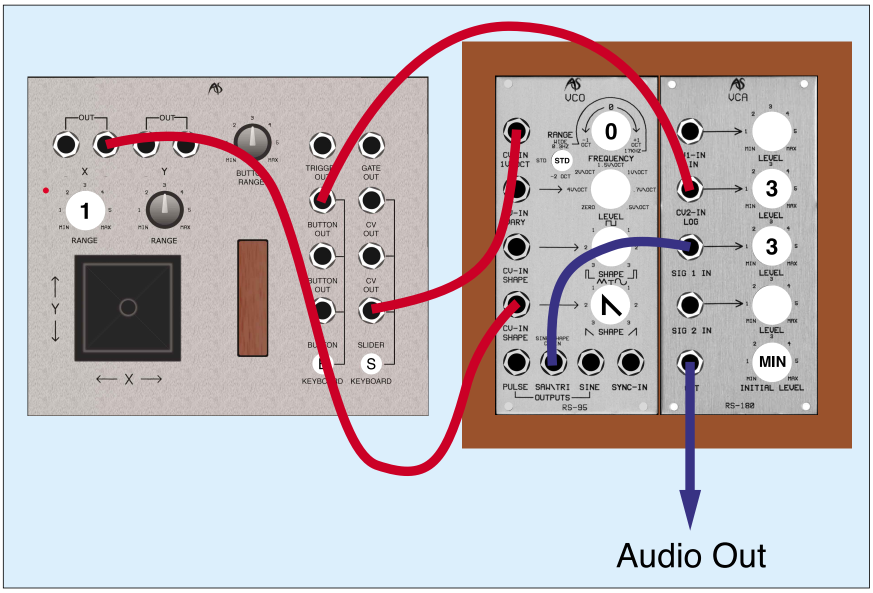


Figure 8: Controlling the harmonic content of the sound.

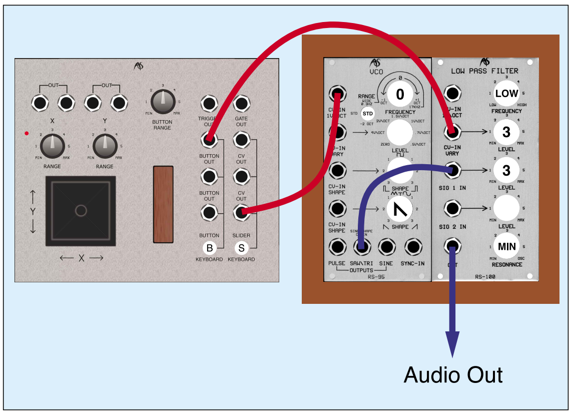


Figure 9: A simple but impressive (and, with the correct articulation controllers, *expressive*) brass patch.

a string instrument? Well, yes, I have. However, the ability to *play* the synth in the manner of a stringed instrument overcomes so many of the limitations imposed by a CV/Gate keyboard that we no longer need many of these to create recognisable imitations and performances. And they're *better* performances, believe me.

## Adding Articulation To Other Sounds

Once you've got the hang of shaping notes and pitching them using the Ondes Martenot architecture, there's no reason to confine yourself to imitations of violins and cellos. You can create stunning imitations of instruments such as flutes and, in particular, vocal 'formant' sounds (both of which will be the subject of a future Synth Secrets). As for generating new sounds and effects, the freedom afforded by the ring and button opens up completely new areas of sound design and creation.

But perhaps the most fun (for today, at least) are the brass sounds that you can conjure effortlessly by replacing the VCA in Figure 7 with a low-pass VCF (see Figure 9 above).

As you can see, we're still using just two modules, and the patching is identical, but the button — instead of controlling the loudness — is now controlling the cutoff frequency of the filter, and therefore the tone of the sound. And, because the initial cutoff frequency is set to Low, all the harmonics are filtered out until you press the button, which means that silence reigns between notes. Consequently, the filter is not only shaping the tone of the sound, it's also differentiating one note from the next. This is incredibly elegant!

If you now play a note, articulate it with the button, and add suitable vibrato using the ring/wire, the result is magic, especially when played through a good digital reverb. You can play distinct notes, imitate swell, recreate the mis-pitching of certain notes that invariably occurs when playing real brass instruments, slide notes to imitate trombones... and if you detune the oscillator by a couple of octaves, you'll obtain the most realistic tubas you've ever heard from an analogue synth.

So there we have this month's Synth Secret; two modules and a more appropriate method of controlling them can be far more expressive and create more realistic bowed string and brass sounds than any number of modules and facilities controlled by a less suitable device. It's an important lesson, but because of the ubiquity of the keyboard synthesizer, it's not one that many people have had the opportunity to learn. SOS

# Synth Secrets

**The characteristic sound of flute-like instruments is complex — but fortunately not so complex that it can't be emulated fairly successfully with a synthesizer...**

## Synthesizing Pan Pipes

Gordon Reid

Before I ever touched a synthesizer, I played organs. Big ones, mind you... three-manual jobbies with 32-foot pipes that made a sound that you heard with your lungs, not your ears. And on these organs, there were numerous stops called 'flutes'. They sounded nothing like the instruments played by Ian Anderson and James Galway, but produced a softer sound than the strident brass stops, and a rounder one than the nasal reeds. With a following wind and a vivid imagination, you could picture them fulfilling a similar function to orchestral woodwind.

Later on, I bought my first synth and spent endless happy hours setting up the patches I found on the back of its four-page manual. One of these was, inevitably, a flute, but I could never get it to sound like the real instrument. As for the lovely, breathy sound of the pan flute, I couldn't come close. This was very disappointing, but I persevered because I knew that synths could produce good imitations of flutes... I had seen Genesis, and watched Tony Banks use an ARP ProSoloist to play Peter Gabriel's flute parts to remarkably good effect.

Nonetheless, my Korg 700, its replacement the MS20, and later additions such as the Roland SH1000 and SH2000 all failed to deliver, and it wasn't until I bought some rather sophisticated synths in the mid-'80s that I managed to conquer these sounds. So, in an effort to save you from my decade of frustration, we'll embark upon the Synth Secrets guide to synthesizing flutes.

### The Principles Of The Flute

There are many instruments in the flute family, all of which use a sharp edge to excite a column of air in a cylindrical pipe. The family includes pan pipes, recorders, the Shakuhachi, and organ pipes, as well as the familiar orchestral flute and piccolo.

We can summarise the difference between these and brass instruments by referring to Figures 1 and 2, below. The first of these shows a brass player blowing directly into a pipe and establishing a standing wave, the frequency of which is defined by factors such as lip tension, blowing pressure, and the length of the pipe. The second shows a flautist exciting a column of air by blowing against the edge of a hole in a pipe.

You may think that these cases are similar to one another, but they are not. The physics of the directly blown pipe is quite different from the fluid dynamics and aerodynamics needed to understand and explain the flute. Consider the pitch and tonal changes available from over-blowing a flute, and the manner in which a skilled player can change the tone by altering aspects of the blowing angle and pressure, and it is apparent that something very complex is happening. What's more, it isn't even clear how the act of blowing across the gap creates a musical note. So let's simplify things by considering one of the earliest and most basic of flutes: a single pan pipe.

### A Single Pan Pipe

Imagine you are blowing across the top of a bottle. Experience tells you that, if you get the blowing angle and pressure just right, you'll produce a breathy note that gets deeper in pitch the larger the bottle happens to be. The pan pipe — an instrument made from a cylindrical piece of bamboo and tuned to a particular pitch using a wax plug in the bottom — is the same. If you blow at the correct angle across the mouth of the pipe, you create what's called a 'flow valve'. Like a physical valve, this controls how the air flows into or out of the aperture.

Confused? Then imagine that there's already a standing wave in the pipe. At some point in time, the air at the mouth of the pipe is rarefied, and sucks in the air that

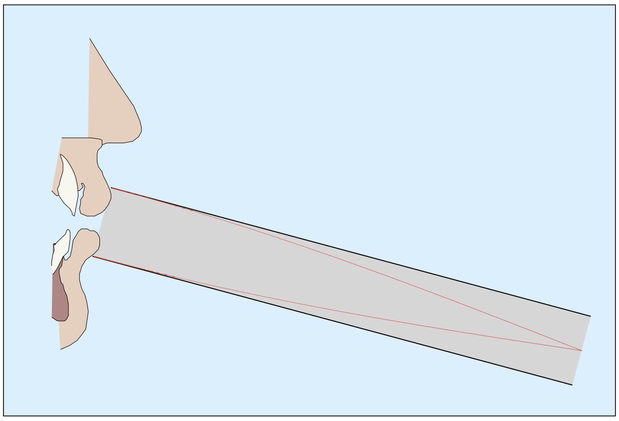


Figure 1: Energising a column of air in a brass instrument.

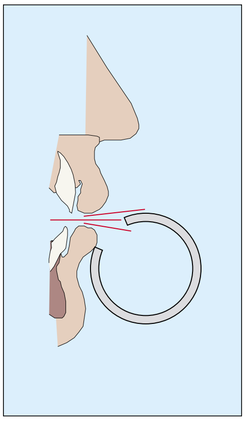


Figure 2: Energising the air within a flute.

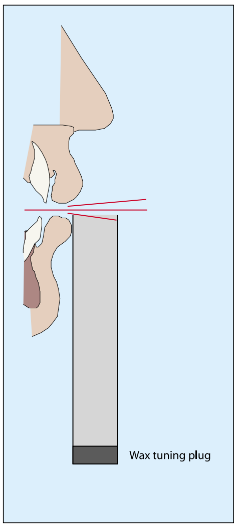


Figure 3: Blowing a pan pipe.

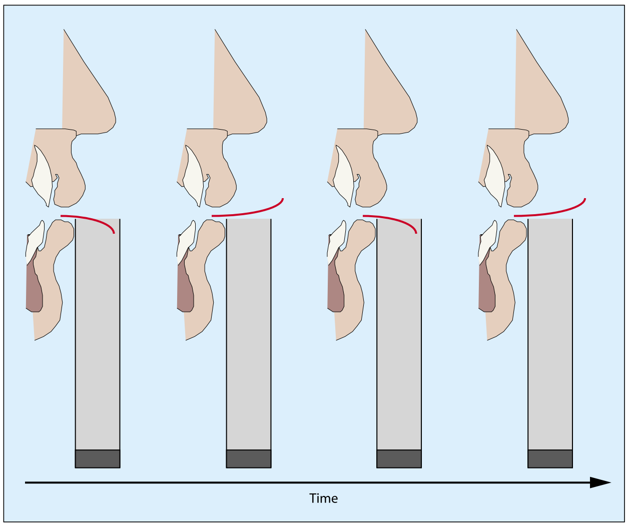


Figure 4: Energising the pan pipe.

the player is blowing across the top. A moment later, the compressed air at the mouth pushes outward, deflecting the player's breath. A moment after that, the cycle begins again. When the length and breadth of the pipe, and the angle and flow rate of the player's breath, are correct, the standing wave will form quickly, and it will exist as long as the conditions remain satisfied. (See Figure 4, above.)

To be honest, this is a simplified explanation, but it's satisfactory if we don't inspect the physics too closely. It then becomes simple to determine the pitch and tone of the note produced. Like the brass instrument in Figure 1, the pan pipe is closed at one end and open at the other, so it supports a standing wave with a fundamental wavelength twice the length of the pipe. However, whereas the brass instrument is closed by the player's lips and open at the far end of the pipe, the pan pipe is open at the energising end, and closed at the bottom. The waveform is therefore inverted, as shown in Figure 5, below.

Although this reversal of the open and closed ends makes no difference to the pitch of the instrument, the shape of the pipe differentiates it from any brass instrument and, as we shall see in the coming months, from some other woodwind instruments. The difference is this: a cylindrical bore closed at one end (a pan pipe) can produce only odd harmonics (see Figures 6 and 7, below). In contrast, a conical bore (brass) produces a full harmonic series. This means that the pan pipe shares its tonality with the family of waveforms that includes triangle waves and square waves whereas, as we have seen before, brass is better synthesized using the sawtooth waveform.

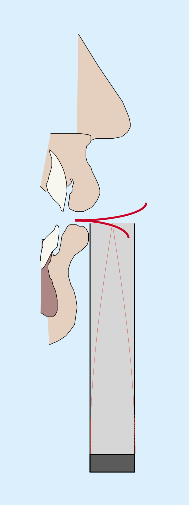
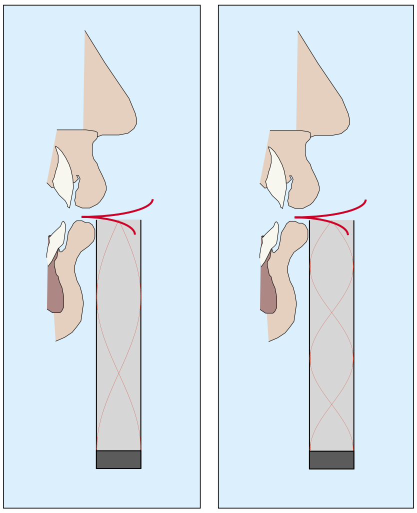


Figure 5: The fundamental of a pan pipe.

Inevitably, things are not as simple as this because — as I've mentioned before when we've discussed pipes — the wavefront overshoots the end of the pipe by a small distance, so higher modes of vibration become progressively inharmonic. What's more, you can affect the nature of the vibration by covering a proportion of the aperture (which allows the player to bend notes) and by changing the air flow (which makes possible a wide range of breathy timbres, and the instrument's characteristic percussive sounds).



Figures 6 and 7: The 3rd and 5th harmonics generated within a pan pipe.

## Enough Of The Theory... Yes?

Despite these complications, it seems that the pan pipe is, essentially, a generator of square waves or something similar, so we should be able to imitate it using the simple synthesiser architecture shown in Figure 8, below. Unfortunately, this doesn't work. The result sounds like a contoured square wave, and nothing like the real instrument.

If you listen to a pan pipe (or, for that matter, a blown bottle) the major reason for Figure 8's timbral inadequacy is obvious. Despite the seeming thoroughness of the analysis above, the real sound has a very strong noise component. This is a consequence of turbulence.

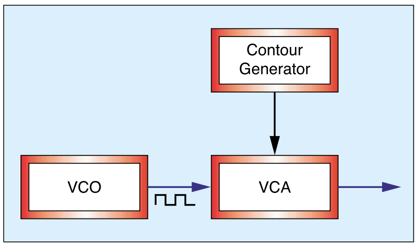


Figure 8: A simple synthesizer for generating sounds based on square waves.

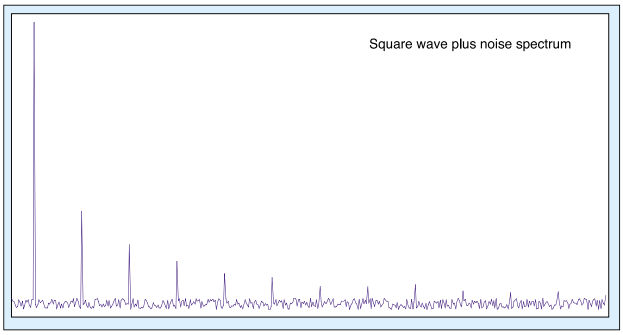


Figure 9: A spectrum resulting from adding low amplitude white noise to a square wave.

Let's digress for a moment, and consider one of those TV adverts from the 1990s that showed the air flowing over the body of a car. As the adverts explained, a smoother air flow offers benefits such as improved fuel economy and  lower noise in the cabin. If anything disturbs this ideal flow, turbulent vortices appear. These create drag that slows the vehicle, make it less fuel efficient, and make the cabin considerably noisier.

Even a small amount of turbulence can

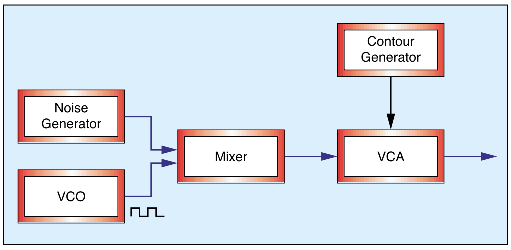


Figure 10: Mixing a square wave and noise to try to generate the pan pipe sound.

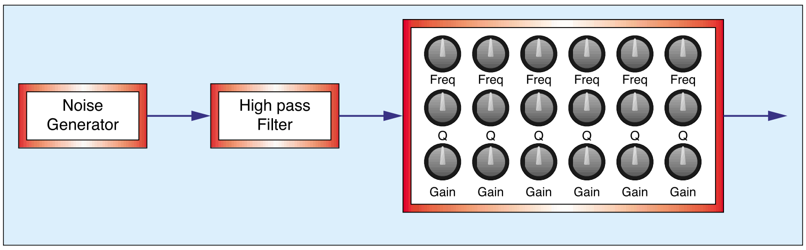


Figure 12: Tuning noise to the square wave harmonics.

generate a considerable amount of acoustic noise, as you will appreciate if you stand underneath a jet aircraft as it takes off. Likewise, the turbulence in a musical instrument will add a strong noise component to its sound. Not surprisingly, given our observations of the real instrument, the excitation method — blowing against an edge — is a superb mechanism for generating turbulence within and outside the pipe, so we must expect the resulting spectrum to comprise a square wave plus noise, as shown in Figure 9 on the previous page.

Nevertheless, adding a noise generator to Figure 8 (see Figure 10, above) proves no more satisfying than the earlier patch. The square wave and the noise seem to be disassociated from one another, and this sounds wrong.

Listen again to the pan pipe, and you will hear that the tonal part of the sound and the noise are not independent of one another. The noise has a breathy quality with a distinct pitch related to the note being played. This is not surprising. The turbulence occurs within the pipe and at its boundaries, so it must be coloured by the acoustics of the pipe itself.

Listening even more closely, it's apparent that the tonal part of the sound is not rich in high-frequency harmonics. In contrast, the noise is most audible at higher frequencies. This means that we have a spectrum that is more like Figure 11, below, with strong, low harmonics accompanied by a halo of noise, and higher harmonics that are masked by broad, noisy bands of frequencies.

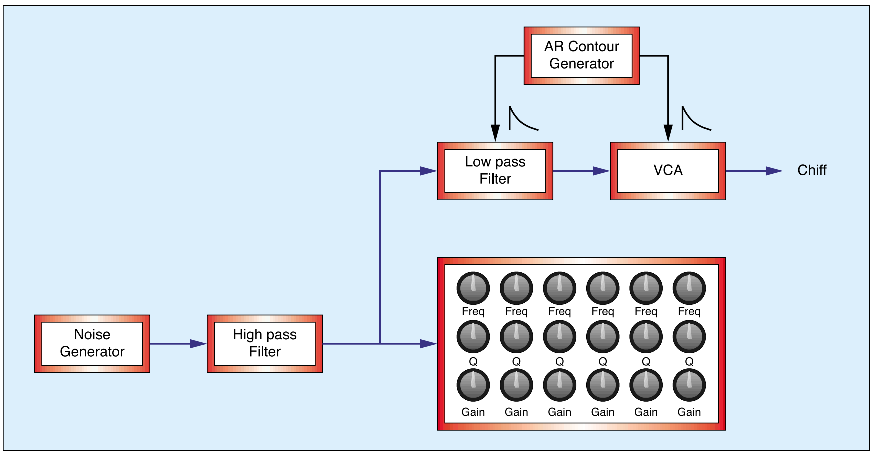


Figure 13: Patching a 'chiff'.

We can synthesize this. Although few, if any, subtractive synthesizers offer 'blue' noise sources (in which high frequencies predominate) it's easy to patch this: send a white noise source through a gentle high–pass filter. You can then pass the result through a formant filter bank tuned to the harmonic frequencies of the note produced by the pipe. Hmm... this is non-trivial. Apart from the tuning itself, we will need as many formants as the square wave has harmonics. For a note close to middle 'C', we need filters for approximately 250Hz (the fundamental), 750Hz, 1,250Hz... and so on up to 20,000Hz. That's 40 band-pass filters! Fortunately, experience shows that just six bands on the edge of self-oscillation, tuned to octaves and fifths, provide an excellent 'breathy' sound that fulfils our purposes admirably. I have drawn this in Figure 12 at the top of the page. Now let's listen to the real thing yet again, and try to determine how the parts of the sound are developing...

The first thing we hear is a noisy 'chiff' that sounds independent of the tone and the breathy noise that we just created. Skilled pan pipe players make great use of this, and it is perhaps the most defining characteristic of the instrument. Consequently, we need to add a chiff. We do so by tapping the filtered noise before it reaches the formant filter, and passing this through a low-pass VCF and a VCA, both of which are controlled by an AR contour generator. (See Figure 13, above.)

After the chiff, the noise settles into its quasi-tonal form, so we need to control the amplitude of the output from the formant filter bank, and mix this with the chiff. Ideally, one would do this using an independent VCA and another contour generator, but in the interest of simplicity I am going to use a single ADSR contour generator as patched in Figure 14, overleaf. This works because I can set the cutoff frequency of the 'chiff' filter to pass signal only during the A and D phases of the contour. Neat, huh?

## Set The Right Tone

Now let's return to the tonal element of the sound. We know that this has only odd harmonics, so we need to start patching using a square wave or triangle wave as the oscillator's output. We also know that we must filter the harmonic content to be akin to that of the real pan pipe. This means that it must be fairly mellow, so we place a low–pass filter in this signal path.

Listening to the original instrument yet again, you'll notice that the tonal sound does not begin immediately, but swells up after the chiff and after the tuned noise becomes apparent. We could imitate this perfectly with one of Korg's HADSR (Hold-

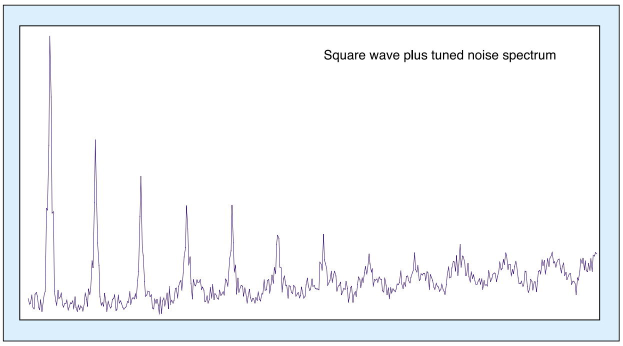


Figure 11: Turbulent noise tuned to the harmonics of the pipe.

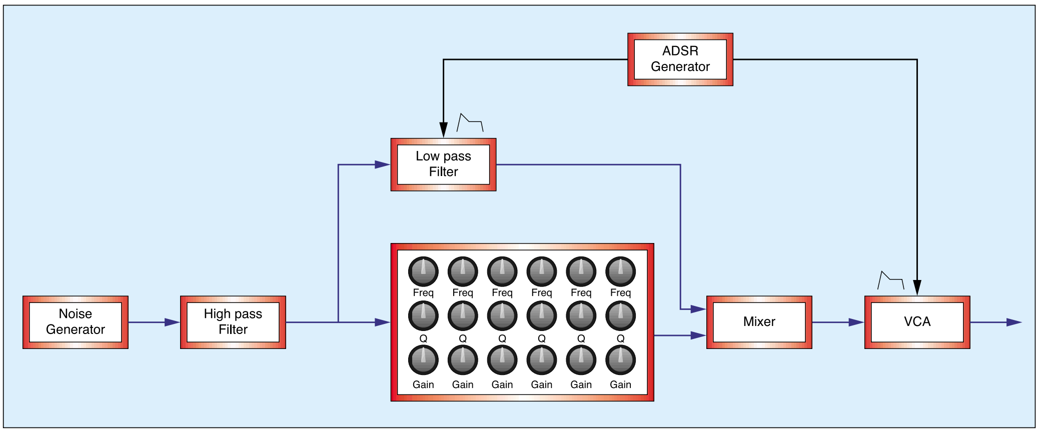


Figure 14: Creating the chiff and breathy elements within the sound.

Attack-Decay-Sustain-Release) envelopes, inserting a tiny delay before the onset of the Attack phase. Unfortunately, few synths have these envelopes, but we can get away with a simple ADSR if the Attack value is carefully chosen. Figure 15, below, shows how the two contours combine the wave and the noise into the composite sound.

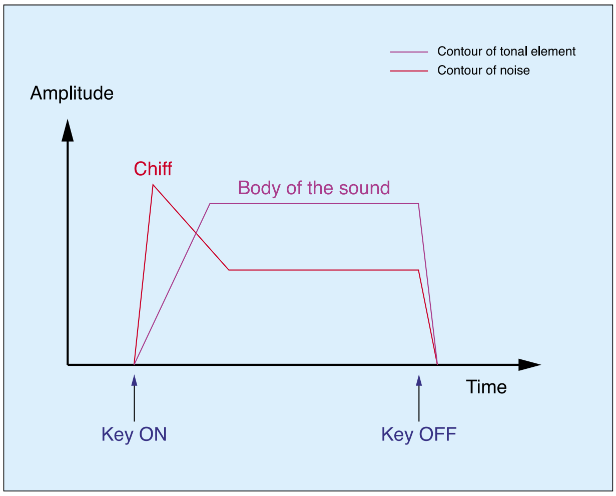


Figure 15: The contours for the noise signal and the pitched signal.

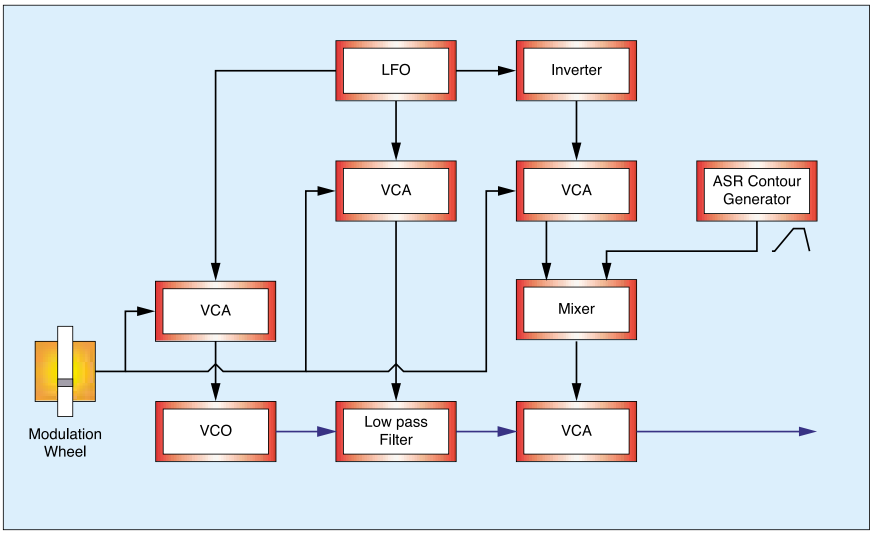


Figure 16: Shaping and modulating the tonal elements of the sound.

Refining the pitched sound still further, a bit of modulation wouldn't go amiss, so we'll add an LFO to create vibrato, and then a wheel to control its depth. In fact, let's go the whole hog, and patch some vibrato, tremolo and a little bit of filter modulation simultaneously. But, contrary to everything we have learned about natural sounds, we'll invert the amplitude modulation so that, as the filter opens, the gain is reduced, and vice–versa. This seems odd, and I wouldn't have tried it had fellow *SOS* contributor Nick Magnus not bullied me into it. Nevertheless, it seems to work well, keeping the total amplitude steady as the filter opens and closes. I have shown all of this — the filtered square wave, the envelope, and the modulation — in Figure 16, at the bottom of the page.

Figures 14 and 16 contain almost everything we need to produce the single note produced by one pan pipe. However, a single note is not of much use unless you're into minimalist Andean avant-garde music. So we need to add some control signals that will make the patch work over a range of notes. We'll provide these from a conventional CV+Gate keyboard, but keep the following points in mind as we patch it in.

Firstly, it's vital that the oscillator in Figure 16 and the formant filters in Figure 14 track the pitch CV together. If they do not, the two elements of the sound will disassociate, and ruin the illusion.

Secondly, it's important that the keyboard offers multi-triggering. This ensures that the chiff occurs at the start of every note, even when you play legato.

Thirdly, I'm going to add an attenuated pitch CV to open the filter in the lower signal path, allowing us to make the signal brighter as the pitch rises, but not necessarily in a 1:1 relationship. This is, of course, variable keyboard tracking.

Next comes something we've not tried before in Synth Secrets, and I'm again indebted to Nick Magnus for suggesting it. If you refer back to Figures 9 and 10, you'll remember that the oscillator signal and the noise signal failed to form a composite sound. We overcame this in Figure 14 by tuning the noise to the harmonics of the square wave, but we can do even better.

Applying noise to the CV input of the low-pass filter shaping the square wave signal adds a rough edge to the sound. It's noise, but with a very different character to that obtained by adding audio-signal noise using a mixer. What's more, you can manipulate the tone and amount of this noise using a graphic EQ or 'fixed filter bank', so that it sculpts the sound in desirable ways. If you make sure that the noise in this part of the patch is at predominantly high frequencies, and apply just a little to the filter CV input, it works a treat.

Right... now we're ready for Figure 17, over the page.

Despite these refinements, the sound is still somewhat artificial in nature. With a MIDI keyboard that accepts a breath controller, plus a suitable MIDI/CV converter, you could animate the patch in ways that are not possible with envelope

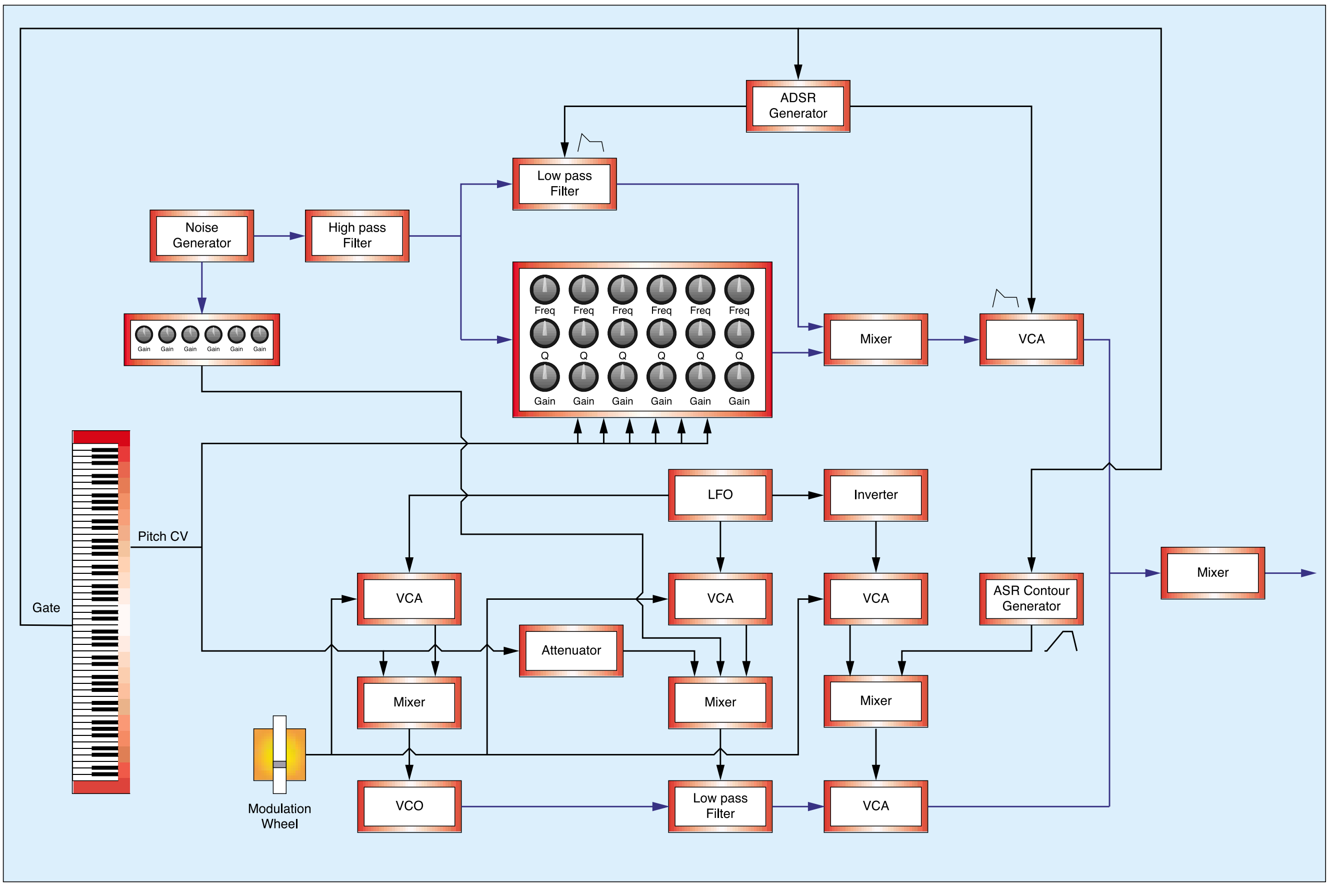


Figure 17: The pan pipe patch.

generators and LFOs. You could improve matters even further by replacing the keyboard, modulation wheel and breath controller with the Ondes Martenot discussed last month. What do you mean, you don't have an Ondes Martenot? Oh well, there's another good solution...

In the hands of a skilled player, a real pan pipe exhibits a great deal of pitch bend, as well as vibrato and tremolo. So — instead of using an Ondes Martenot or breath controller — our final development involves replacing the modulation wheel with an X/Y joystick, and patching it so that the Y axis provides control over modulation depth and the X axis provides pitch bend. For clarity, I have shown the relevant part of the patch in Figure 18, below, and incorporated it into the final diagram, opposite, as Figure 19.

If you've set the controls of each module appropriately, this patch now sounds very much like a pan pipe. In fact, it sounds more like a pan pipe than I would have thought possible before I developed it. It's true... Figure 19 is not just theory; I created it using one of my analogue modular synth systems, and it sounds superb.

But you can go still further. For example, if you're programming on a modern computer-based software-synth package such as NI's *Reaktor*, or a hardware/software system like Clavia's Nord Micro Modular, you can make the attack velocity sensitive, which proves to be another huge improvement. You could also patch the joystick's pitch bend to amplitude, and there are other flourishes that would add performance and realism to the sound. But, however you choose to complete the patch, just add a judicious sprinkling of reverb and... *hola*, *amigos*!

## Epilogue

I have looked through the patch

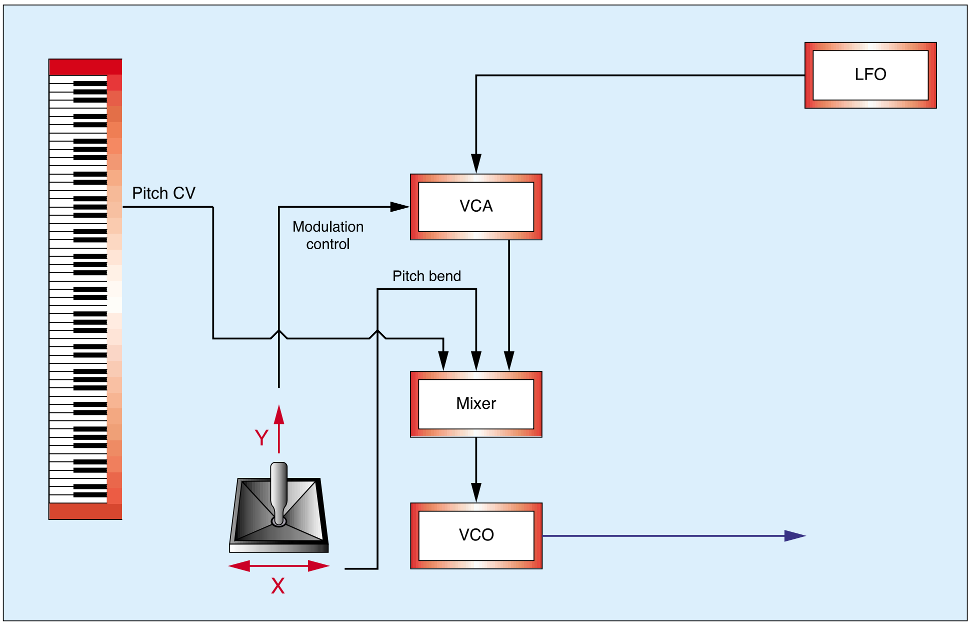


Figure 18: Using a joystick for added realism.

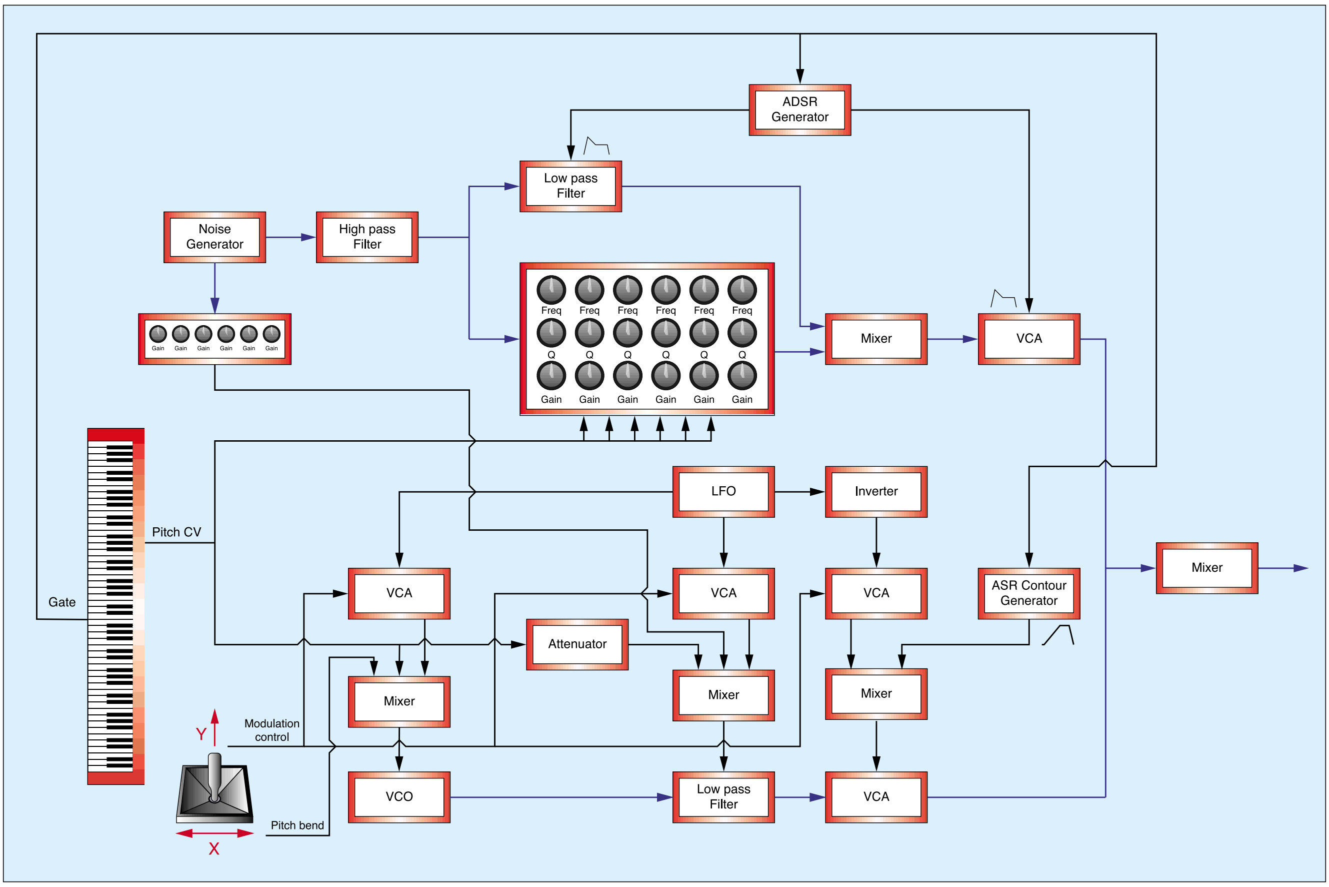


Figure 19: One more time, with feeling.

books supplied by ARP, Moog, Roland, Korg and others, and despite a wealth of flute patches, I could find no pan pipes. Yet here we have a remarkable patch that requires just four voltage controlled filters, a formant filter, a fixed filter bank, an oscillator, a noise source, seven VCAs, four contour generators, a bunch of mixers and multiples (which I haven't even shown), a joystick, and... Ah yes, I see the point. Pan pipes may be straightforward to synthesize in software or on something the size of a small wardrobe, but their instantly recognisable 'breathy' sound is not going to emerge unscathed from a Minimoog, Odyssey or SH101. Nonetheless, orchestral flutes pour forth from basic synths. Despite the increased mechanical complexity of the flute, its sound must be simpler than its predecessor, the pan pipe. So, next month, we'll create some patches that you'll be able to try on almost any synth. Until then...

# Synth Secrets

The Monty Python team once famously claimed that being able to play the flute was a simple matter of 'blowing *here*, and moving your hands up and down *here*'. But there's a lot more to it than that...

## Synthesizing Simple Flutes

Gordon Reid

Last month I discussed the sound of the pan flute, leaving you with a diagram that showed how you could use a large modular synth to create a remarkably accurate simulation of the instrument. Successful though it was, the patch was a monster, and not one you could create on any basic (read... easily affordable) synth. This month, we'll look at another of the flute family, and see whether we can synthesize it using something rather simpler.

But first, I want to take a look at the Japanese Shakuhachi. Made from a single piece of bamboo, this is another instrument that requires you to blow over an aperture, but it differs from its more primitive cousin in three ways.

Firstly, you excite the air by blowing over a sharp edge at the mouth of the pipe. If you consider Figure 1 (below) you can see an instance when a jet of air blown against such an edge is deflected downward. At that moment, the part of the stream shown in orange is moving minutely faster than that shown in red, so the air pressure on the flow tends to press it upward. This is the principle that keeps aeroplanes in the sky: air moving faster over the top surface of the wing generates less pressure than that moving more slowly along the shorter underside. The net effect is therefore an upward pressure that lifts the machine off the ground.

If the upward pressure in Figure 1 is sustained for a fraction of a second, the jet is pushed upward, and we soon reach the situation shown in Figure 2. Now, the net atmospheric pressure is downward, and we quickly move back to the situation shown in the first diagram.

If the edge is connected to a pipe of some sort, it doesn't take much of a leap to realise that, at some frequency, the up/down vibration of the airflow will match the pipe's resonant frequency, and a standing wave will result, generating a sustained note. It turns out that the speed of up/down oscillation is roughly proportional to the speed of the air stream (ie. how hard you blow), which explains why, when you blow harder into an instrument of the flute family, the note jumps from the fundamental to the second harmonic, and then the third, and the fourth... and so on, as the increasingly rapid up/down motion excites higher modes of oscillation in the pipe.

The second difference between the pan flute and the Shakuhachi is that the latter is open at the bottom. Therefore, as explained in part 24 of this series (see *SOS* April 2001, or surf to: www.soundonsound.com/sos/apr01/articles/synthsecrets.asp), the standing wave within it contains both odd and even harmonics. Indeed, all the instruments of the flute family are 'open' if they have holes, so the pan flute proves to be an oddity... it's the only flute that generates no even harmonics (this is not strictly true, although it is true for orchestral flutes. There is a class of organ pipes called stopped flutes that are closed at the top using wooden plugs. These generate the odd-harmonic series discussed last month, and offer a significant advantage to the organ builder; for any given pitch, they need only be half the length of their open brethren. Given that the longest open pipes are typically 32' long, this means that an organ with stopped 16' pipes can produce the same, deep pitches as larger instruments).

Thirdly, the Shakuhachi has holes. All the pipes we have considered before — whether open at one end or both, whether cylindrical or conical, and whether made from wood or brass — have boasted continuous bores. Even the valves on instruments such as the

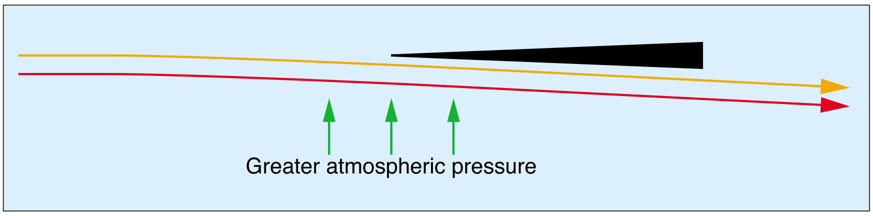


Figure 1: The pressure exerted when air passes under a sharp edge placed in the stream.

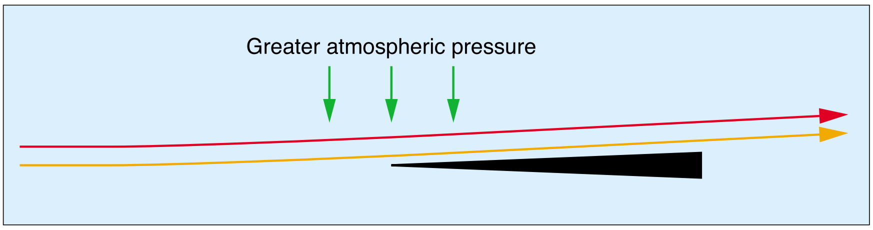


Figure 2: The pressure exerted when air passes over a sharp edge placed in the stream.

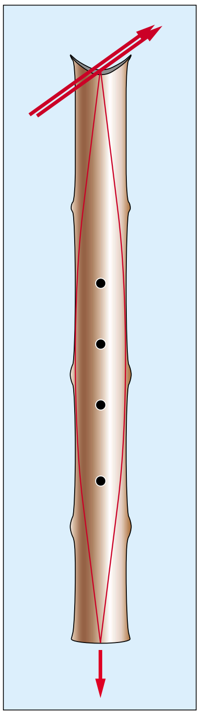


Figure 3: Blowing a note on the Shakuhachi with all holes closed.

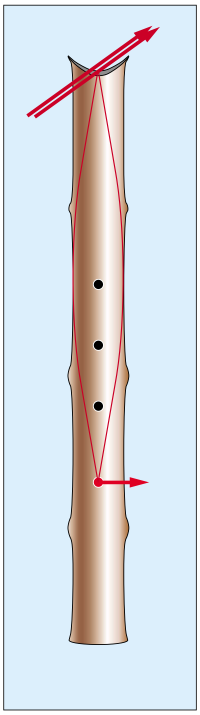


Figure 4: Blowing a note on the Shakuhachi with all but the lowest hole closed.

trumpet do not change this: they alter the length of the pipe, but they don't allow air to escape before the end is reached. Sure, there have been all sorts of complications such as end effects and harmonic stretching, but when it comes down to it, open pipes of a given length have always had a fundamental frequency of a certain pitch, (or an octave lower if you consider their closed brethren).

The holes complicate matters considerably, but for now we'll consider them simply to be ways of shortening or extending the effective length of the pipe. If you look at Figure 3 (above), you can see that, with all holes closed, the Shakuhachi produces the pitch associated with its entire length. But when you open the bottom hole (as shown in Figure 4), the effective length becomes shorter, a standing wave of shorter wavelength is generated, and a higher pitched note is produced. Figure 5 (right) then shows what happens when you open the next hole... and so on. Given that the Shakuhachi has five holes — the four that you can see in these diagrams, plus a thumbhole that you cannot — it's not surprising that the instrument is ideal for playing the pentatonic scale that characterises traditional Japanese music.

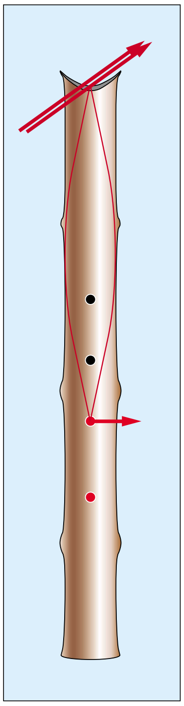


Figure 5: Blowing a note on the Shakuhachi with all but the lowest two holes closed.

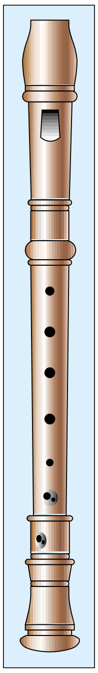


Figure 6: The recorder.

## The Recorder

In the western world, the most common flute with holes is probably the recorder (see Figure 6, right). There are many members of this branch of the family, some with cylindrical bores and some with a combination of cylindrical and truncated conical bores, but all excite the air by passing it over a sharp edge, as shown in Figures 1 and 2.

The shape of the recorders' bore is no accident. A few pages of physics (which, thankfully, we will not reproduce here) show that, if you want to play chromatic music over more than one octave using pipes with holes in the side, only cylinders and conical sections will work correctly. But if the recorder shares with most brass instruments its truncated conical bore and full harmonic series, why doesn't it sound like a trumpet? Clearly, the different excitation methods — lip valve versus edge — must have an effect, and it's no surprise that the recorder lacks the 'parp' of the brass instruments. But perhaps more significant is the fact that the air inside the recorder is excited at its widest point, and the bore then tapers towards the end, whereas you excite a brass instrument at its narrowest point, and the air column then flares towards the end. What's more, brass instruments have a horn that stretches the frequencies of the harmonic series considerably, and recorders do not.

All recorders share the same fingering system: six primary holes, a thumbhole underneath the pipe, and an additional 'little finger' hole at the far end. You might think that this would restrict the number of pitches available, but it turns out that every semitone is available, although the tuning is stretched by more than a semitone from the lowest note to the highest. Fortunately, it is possible for the player to correct this (or at least reduce its effect) by blowing low notes more strongly that high ones.

If you have played a recorder (and who, in an English primary school, ever escaped?) you will know that there can be numerous ways to finger a given note. But if this is true, the holes are not the simple features that we just discussed. Consider Figures 7 to 9 (below), which, for convenience, ignore the conical section within the instrument. These show three notes as played on the common descant recorder. The first diagram shows the instrument with all holes closed, playing the lowest pitch that it can produce. In this case, it's a low 'C'.

The next diagram (Figure 8) shows the same recorder with the lowest hole (ie. the one furthest from the mouthpiece) open. Like the Shakuhachi, this has the effect of shortening the pipe, resulting in the instrument playing a pitch somewhat higher than that in Figure 7. In fact, the note is a 'D'.

But what's happened to 'C#'? Clearly, you cannot play all 12 semitones on an instrument with just eight holes simply by lifting your fingers progressively toward the

▶

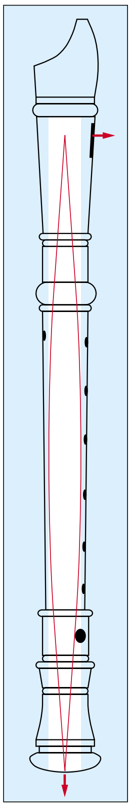


Figure 7: Playing a low 'C' on the recorder.

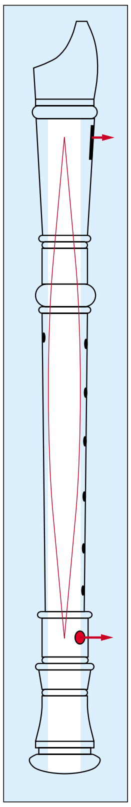


Figure 8: Playing a low 'D'.

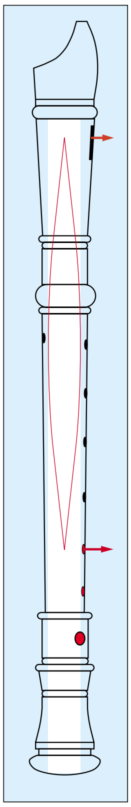


Figure 9: A German recorder's fingering for a low 'F'.

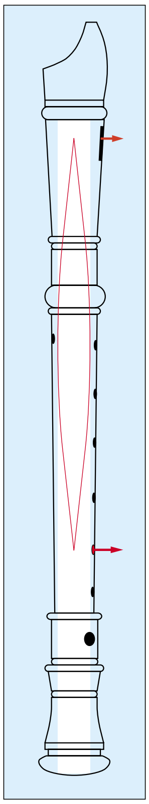


Figure 10: The Baroque recorder's fingering for a low 'F'.

mouthpiece. Nonetheless, the recorder is capable of reproducing the complete scale. The reasons for this lie in some more physics that we'll skip, but which shows that it is not just the position of the holes that change the effective length of pipe; it's also the size of the holes and the amount by which each is covered. The hidden 'C#' is therefore revealed if you 'half-hole' the lowest hole.

Next, we come to Figure 9, which shows the instrument with three holes open. You might expect this to produce an 'F', which it does, if you are playing the 'German' recorder shown in Figure 6. If your instrument is based on the different, so-called Baroque style of recorder, with a different arrangement of large and small holes, the correct fingering is as shown in Figure 10 (above), with the third hole open, but the bottom two closed again. Playing the Baroque 'F' on the German instrument produces a dull 'E'.

This is a very strange result, but it illustrates an important fact: strictly speaking, opening a hole does not 'shorten' the pipe. It acts more like a valve or 'short-circuit' to the outside atmosphere, creating reflections and modifying the wave within the bore. You can have many of these 'short-circuits' along the length of the pipe, and it is their combination and interaction that determine the wavelength and, therefore, the pitch of the note.

If we want to pursue this discussion further, it becomes a bit intense, but the outcome is that there are many ways to combine blowing pressure, aperture size, and closed/open holes to obtain similar pitches on a recorder. Some of these pitches will lie almost exactly on a desired note, while some will be a little sharp, and others will be a little flat. Furthermore, the tonality will differ from one fingering to another, so a skilled player will pick the right fingering according to the demands of the music. This multiplicity of *almost* identical notes leads to a significant problem for the synthesizer programmer; a simple VCO/VCF/VCA patch will never capture the nuances of the instrument. Indeed, this problem is not limited just to the recorder. Many basic wind instruments such as penny whistles, crumhorns, kortholts, rauschpfiefen and cornemuses require alternative fingerings if they are to play a wide range of music. What do you mean, you've no intention of synthesizing crumhorns, kortholts, rauschpfiefen and cornemuses? Oh well...

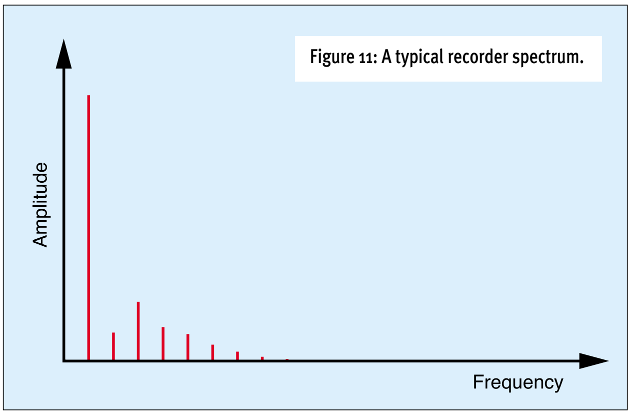


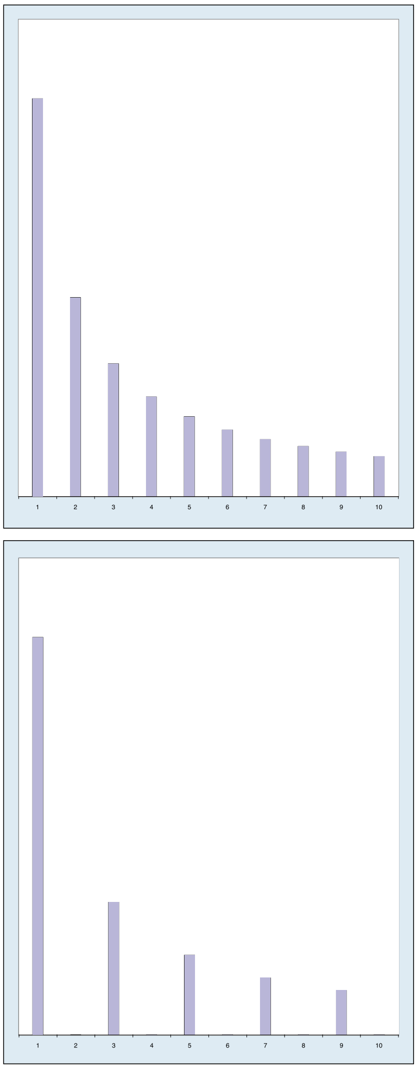
Figure 11: A typical recorder spectrum.

## A Recorder Patch (Take I)

If you trawl through the patchbooks of history, you'll find that almost no synths offer a 'factory' recorder patch. I checked the books supplied with the ARP Odyssey, ARP Axxe, Roland SH101, Korg 700, 700S, 800DV and MS20, and found... nothing. There may be a simple reason for this — that few programmers in the 1970s found the instrument to be very interesting — but I suspect that the true reason is more basic: that it's very difficult to program a convincing recorder patch. Indeed, it's even more difficult than it was to recreate the sound of the pan flute last month. Contrary to most peoples' expectations, the recorder can produce a huge range of timbres, ranging from warm to harsh, from gentle to glassy to brash. Many factors affect this, including the precise size and shape of the bore, the material from which the instrument is constructed, and the quality of the various parts. Some are more suited to solo use, while others work well in ensembles. Nonetheless, I'm not going to wheel out an enormous modular synth to attempt to obtain near-perfection. Instead, I'm going to take the opposite approach, and see whether we can create a reasonable imitation on a much simpler synthesizer.

Let's start by considering the spectrum of a note produced by a typical, modern, wooden recorder (see Figure 11, below left). Omitting from the diagram the strong noise component, this has a dominant fundamental, with a handful of weak overtones. Given that odd and even harmonics are present, we might consider basing our patch on a sawtooth wave, filtering it to attenuate the overtones as shown. But if you try this, you'll find that it doesn't sound right, perhaps because the recorder's second harmonic is so weak. So perhaps a square wave or even a triangle wave (see Figures 12a, 12b and 12c) might sound more appropriate.

In truth, however, none of these sounds right, partly because the spectrum has the wrong shape, and partly because — as we have encountered many times before — the higher harmonics are 'stretched' sharp of their mathematical ideal. Ignoring this stretching (because there's nothing we can do about it) and concentrating on the spectral shape alone, we might get a little



Figures 12 (a) and (b): sawtooth (top) and square wave spectra.

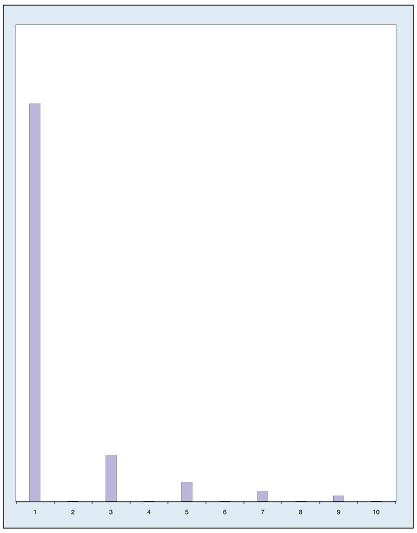


Figure 12 (c): triangle wave spectrum.

closer by summing two oscillators tuned an octave apart. Unfortunately, analogue synths are not stable enough for this to work, and no matter how carefully you adjust the pitches, you'll obtain an inappropriate chorusing of the sound. Digital, additive synthesis would perform better, but that is outside the scope of this month's article. So where do we go from here?

The only analogue synth that I could find which offers a 'factory' recorder patch is the Minimoog, whose programmer, Tom Rhea, based his sound on the oscillator section that I've reproduced as Figure 13 (below). As you can see, Rhea decided that a low-amplitude triangle waveform gave him the closest approximation to the sound he wanted, with just a smidgen of white noise added in the Mixer.

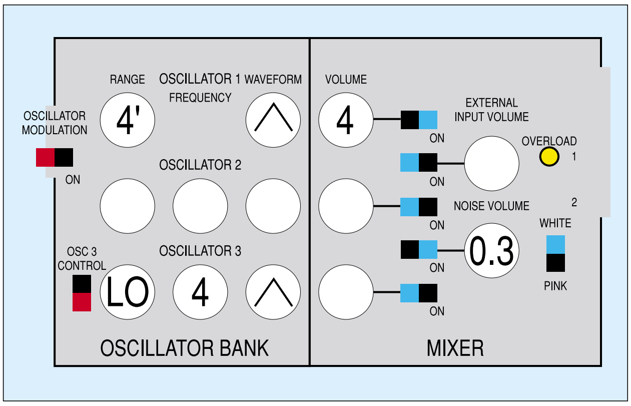


Figure 13: The recorder patch: Minimoog VCOs.

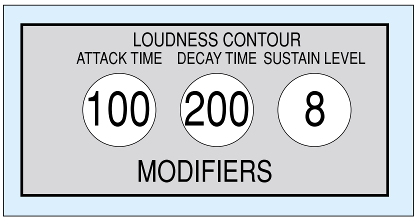


Figure 15: The recorder patch: Minimoog VCA.

The output from the Mixer passes next to the filter, as shown in Figure 14 (below). Note how this is closed until affected by the rapid Attack of the Contour, and that there is a fair mount of Emphasis applied. The uppermost of the three switches is on, so this is where Oscillator 3's output is directed, producing brightness modulation rather than vibrato (if you programme VCO pitch modulation in a recorder patch, it sounds wrong).

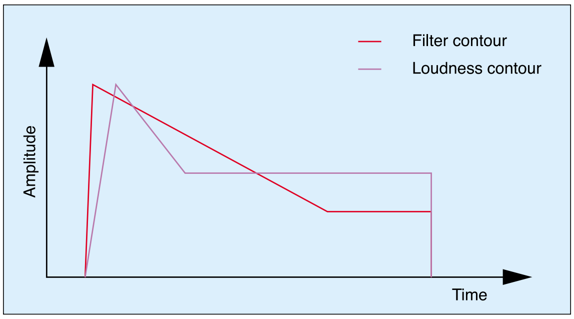


Figure 16: How the contours interact.

Figure 15 (above) shows the amplitude (VCA) contour. The Attack time is rather slower than that of the filter contour, meaning that the brightness of the sound peaks more quickly than the loudness. This relationship is an interesting one, so I have shown it in exaggerated form in Figure 16 (right). This demonstrates that the sound is brightest while it is still getting louder and that, by the time that it reaches maximum loudness, the brightness is already diminished. This lets a small burst of higher-frequency noise through at the start of the note, and goes some way to imitating the sound of blowing the instrument.

If we put all this together, adding the modulation controllers and output section, we obtain the patch shown in Figure 17 (on next page). It will never fool you into thinking that you are playing (or listening to) a real recorder, but with sympathetic performance and careful use of pitch-bend to imitate any changes in blowing pressure, it is somewhat 'recordery'. You can even tune oscillator 1 down to 4' or 8' to imitate alto, tenor and bass recorders, but you must be careful not to play over too wide a range; like the recorder itself, the patch works over two octaves or so, from middle 'C' upwards.

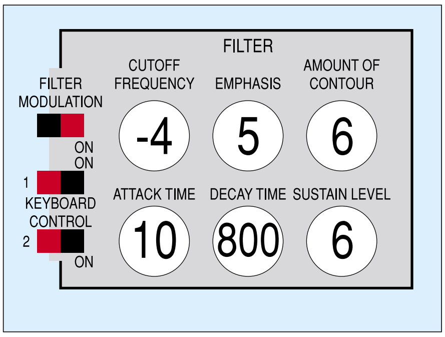


Figure 14: The recorder patch: Minimoog VCF.

## A Recorder Patch (Take II)

For reasons we need not discuss here, I am submitting this article from a hotel room in Tokyo. Given that my studio is somewhat in excess of 6,000 miles to the west of me, I thought that it would be interesting to finish by experimenting with one of the software synths loaded on my G4 Titanium PowerBook to see whether I could get closer to an authentic recorder sound. So I took a trip to Akihabara, bought a Yamaha descant recorder for just 1600 yen (about £8), and used it as the basis for the patch that follows.

Using Gmedia's *Oddity* software (as reviewed in last month's *SOS* — see www.soundonsound.com/sos/aug03/articles/gmediaoddity.htm) that I reviewed last month, I started with an oscillator setting that is not available on the Minimoog, selecting the pulse/square option of VCO2 and setting the pulse width to somewhere in the region of 40 percent. If you followed my explanation of pulse waves and sinc functions a few months ago (see *SOS* March 2003, or link to www.soundonsound.com/sos/mar03/articles/synthsecrets47.asp, you'll recognise that this has a strong fundamental, a weak second harmonic, and a slightly suppressed third harmonic, as required by Figure 11. It retains much of the 'woodiness' of the square wave, but with a bit more 'edge' and, to my ears, is a much better basis for the patch. I set this an octave above middle 'C', made sure that there was no pitch- or pulse-width modulation, and combined the single oscillator with a small amount of

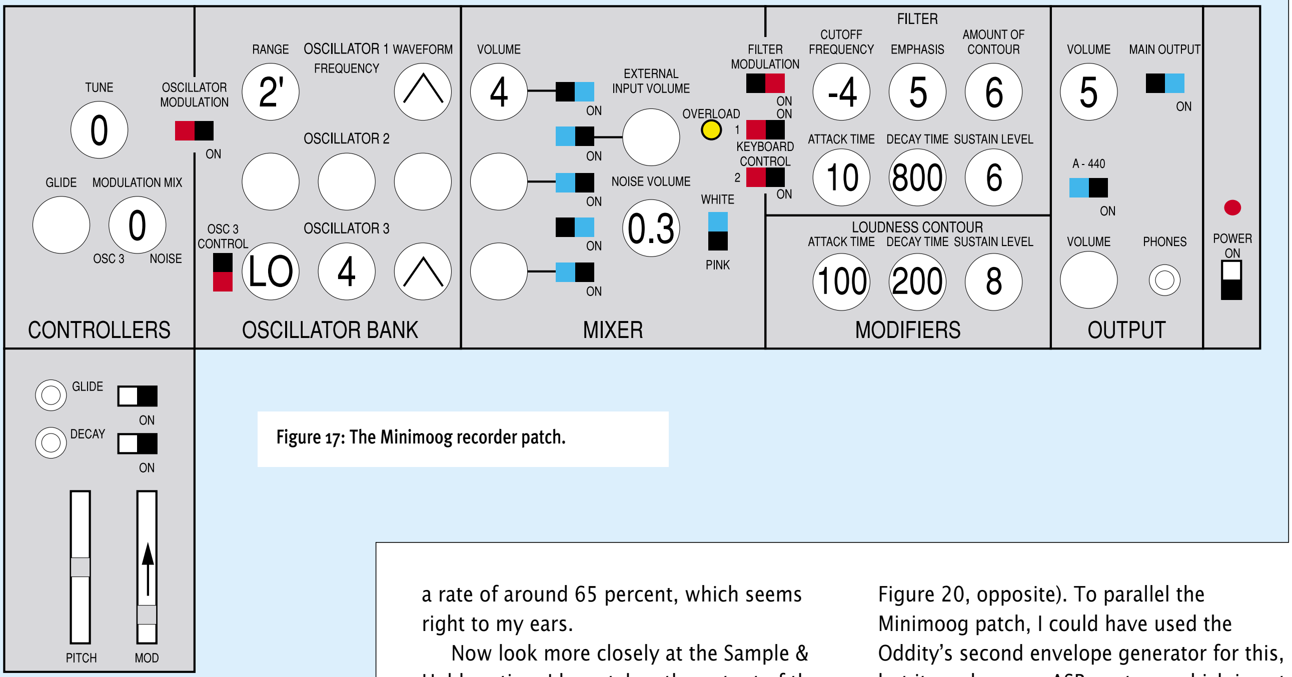


Figure 17: The Minimoog recorder patch.

white noise (see Figure 18, below).

Next, I set the low-pass filter so that, as on the Minimoog, it opened according to the contour determined by the ADSR envelope. I found that, with the release set carefully, I could create a pleasant wooden 'thunk' at the end of the note. I also added a little resonance to add a touch of edginess to the sound (see Figure 19, opposite). Note that the filter tracks the keyboard; in this case at a rate of around 65 percent, which seems right to my ears.

Now look more closely at the Sample & Hold section. I have taken the output of the noise generator (the white slider), sampled it at the LFO rate, and then slewed the result to create a smoothly varying random waveform. I used a tiny amount of this (the yellow slider in the filter section, set to five percent or less) to modulate the VCF. This recreates the small inconsistencies in blowing pressure produced by all but the most experienced recorder players.

Finally, I routed the ADSR to the VCA, setting the initial gain to zero so that no sound leaks through between notes (see Figure 20, opposite). To parallel the Minimoog patch, I could have used the Oddity's second envelope generator for this, but it produces an ASR contour, which is not appropriate for this sound.

The complete patch, shown in Figure 21 (opposite), combines a better approximation to the true waveform of the recorder, an imitation of the instabilities of the instrument, and — in my opinion — somewhat more realistic filtering than we achieved on the Minimoog. But it will never sound convincing, because the recorder has a rather edgy, unstable quality, particularly evident in the flutter that occurs if you blow at a pressure that excites the jet at

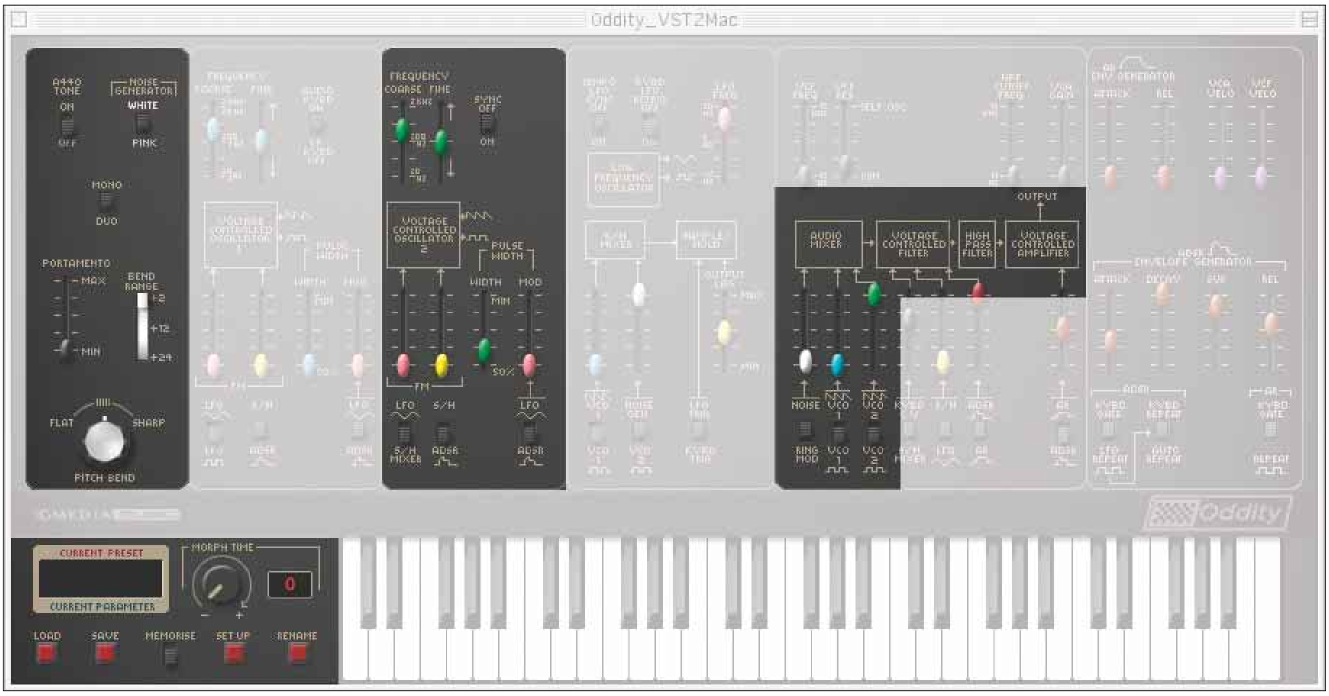
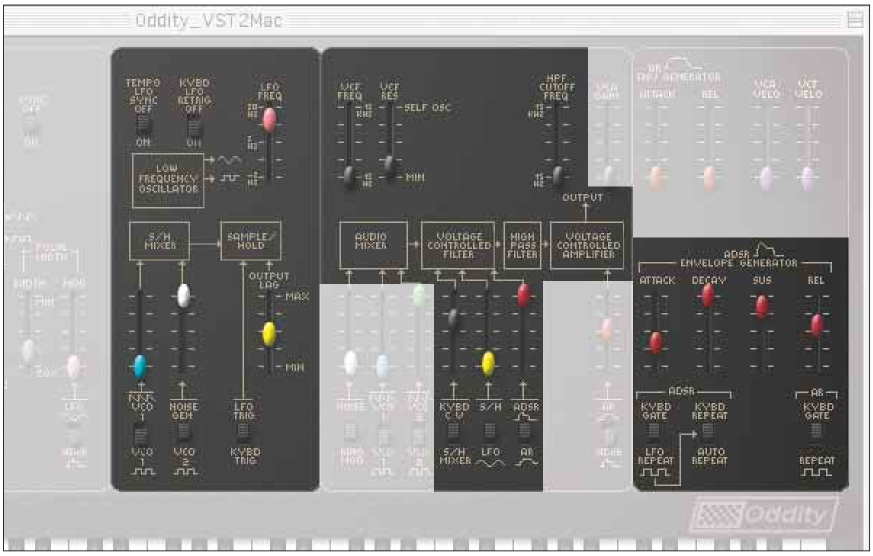


Figure 18: The Oddity recorder: oscillators.

Figure 19: The Oddity recorder: filters.



a frequency somewhere between two of the pipe's modes. The resulting instability, and the jump between modes that occurs if you increase or decrease the pressure just slightly, is the preserve of some very complex patching, or of physical modelling synths such as the Yamaha VL1 or Korg Z1. Furthermore, the recorder produces the tuned noise we discussed at length last month. Clearly, a large, modular synth is going to approach the ideal much more closely than either the Minimoog or the Odyssey.

Furthermore, this Oddity patch lacks the expression obtained by human recorder players who use techniques such as tonguing, gentle attacks, and legato to add interest to their performances. Nevertheless, it has a certain quality that is reminiscent of the original instrument and, with careful playing, it is useable. You can also modify it to coax a range of related sounds from the Oddity, including the aforementioned penny whistles and so on. Sure, it's a delicate timbre that will never grace your dance grooves, but that's not the point. You can learn quite a lot about the strengths and weaknesses of analogue synthesis by trying — and failing — to recreate the sound of the humble descant recorder.

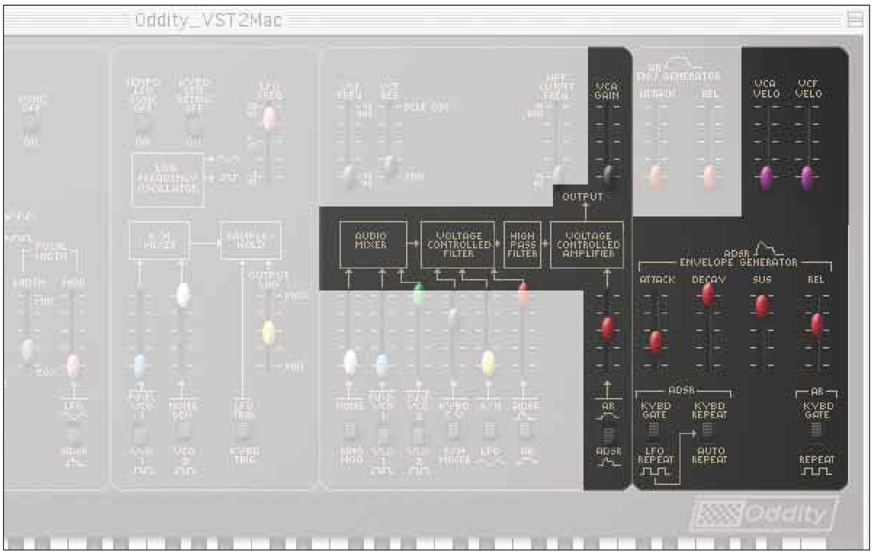


Figure 20: The Oddity recorder: amplifier.

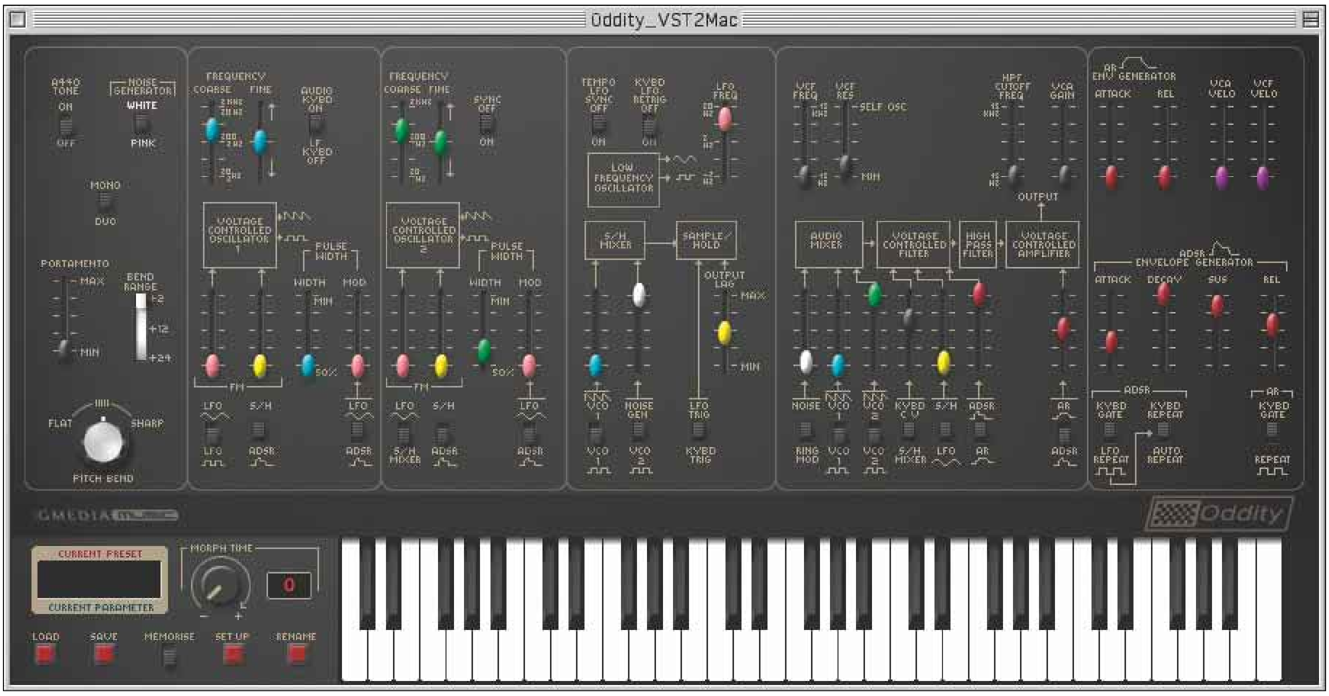


Figure 21: The Oddity recorder: complete.

# Synth Secrets

As we saw last month, there's much to synthesizing a convincing flute sound — and yet basic analogue monosynths have offered reasonable flute patches for 30 years. Surely the process can be simplified?

## Practical Flute Synthesis

Gordon Reid

A couple of months ago, I described and analysed the sound of the pan flute, and at the end left you with a diagram — reproduced below as Figure 1 — which showed, without reference to any particular analogue synth, how you could use such an instrument to create a remarkable simulation of the original pipes. I then told you that I had programmed this to remarkably good effect, but — partly due to space constraints — I didn't show you how.

That omission led some readers to ask to see the patch, so it's shown on the next page, as lovingly crafted on my Analogue Systems Sorceror plus part of the RS Integrator that sits alongside it (see Figure 2). It's possible that I (and indeed you) could create something similar on a huge Moog Modular, a fully populated Roland System 700, a wall-sized Roland System 100M, or a well-endowed Doepfer, but I don't own any of these.

This, then, suggests a problem: given the non-trivial nature of this patch, it's not one that you'll be able to create on a basic analogue synthesizer. Yet, as I stated at the time, good flute sounds (as opposed to pan flute sounds) pour forth from basic monosynths well past their 30th birthdays. Last month, I tried to reduce the complexity (and therefore the cost) of producing a flute-like sound, programming a recorder patch using the *Oddity* software synth loaded on the Apple Power Book on which

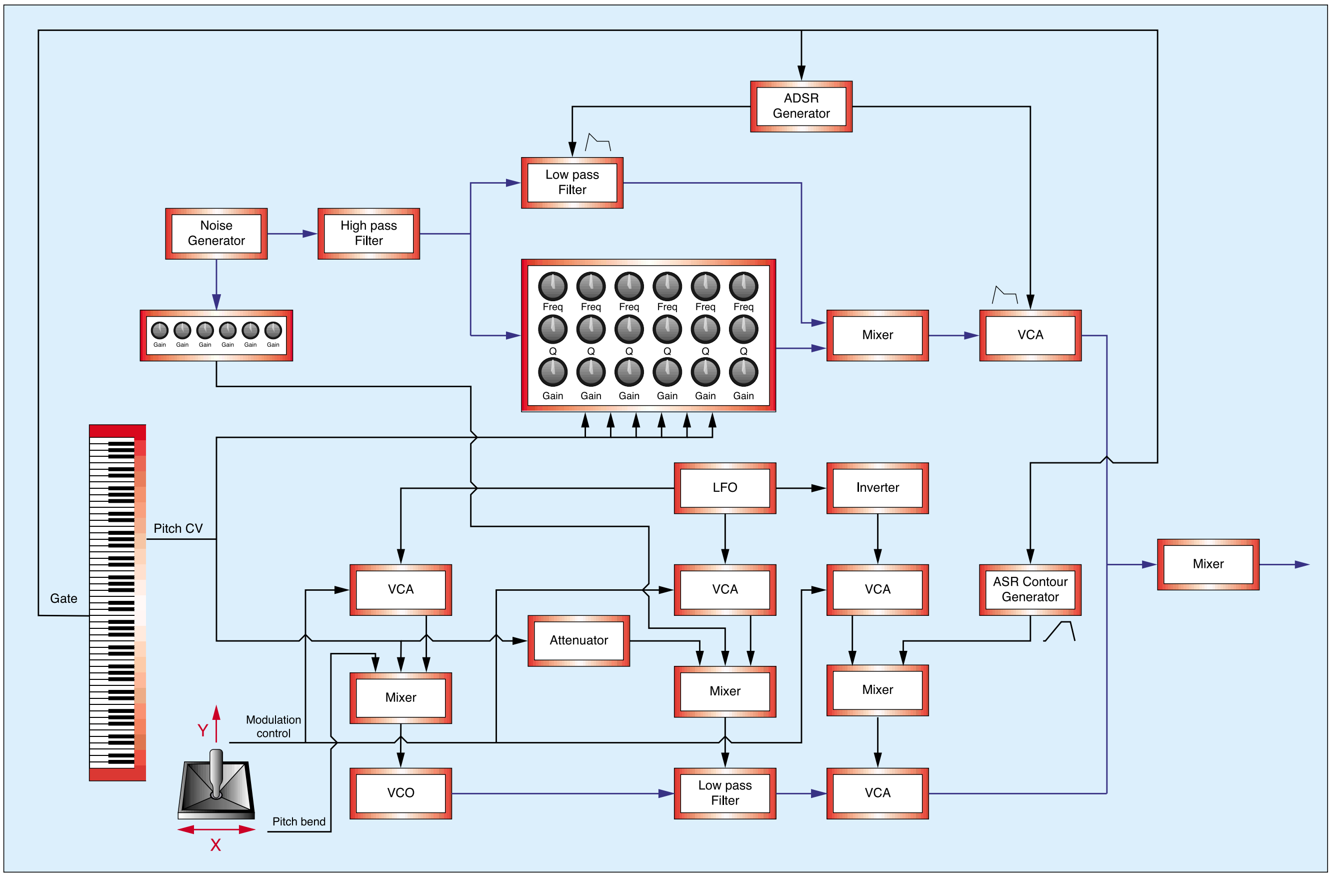


Figure 1: Synthesizing the pan flute.

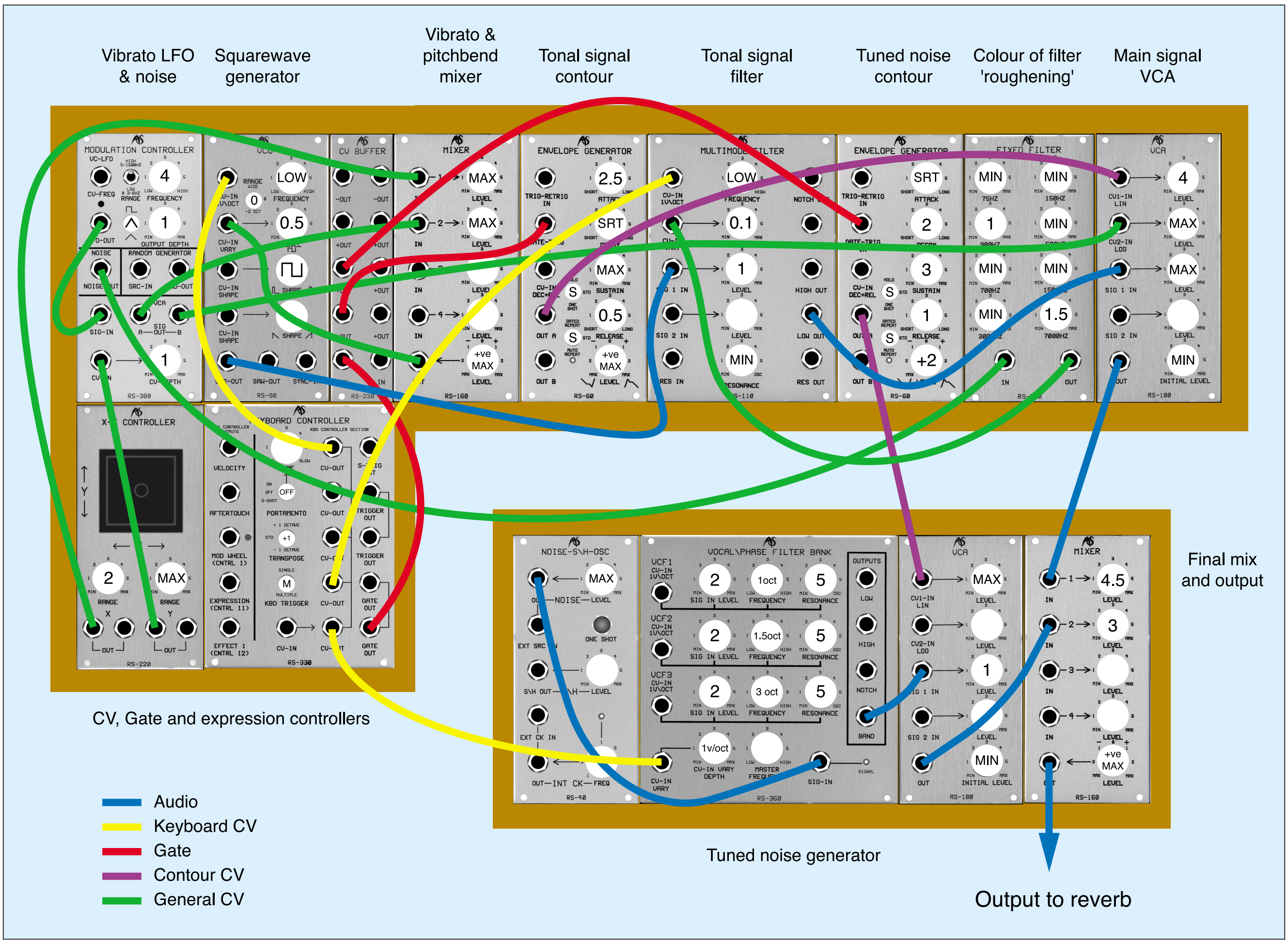


Figure 2: Patching the architecture in Figure 1.

I'm writing this. As expected, the results were far less than convincing, although they were useable in a 1970s sort of way.

But, just to demonstrate that the complexity and expense of a modular synthesizer isn't always necessary to create a superb sound, I'm going to start by telling you that my favourite analogue flute resides within a small synth that is the antithesis of the Sorceror and Integrator shown in Figure 2. Designed to sit on top of an organ, and to be as much at home in tearooms and dance halls as in rock venues, this is the ARP Pro Soloist, which first appeared in 1972. On the surface, this is a simple VCO/VCF/VCA preset synth that offers the player almost no control over the sounds its produces. But if you delve deeper, you'll find that it's a remarkable instrument that, on another day, might command a complete instalment of Synth Secrets. Unfortunately, understanding the Pro Soloist does not further our understanding of the flute, so we must move on without it. That's because it's now time to look more closely at the flute itself.

## The Modern Flute

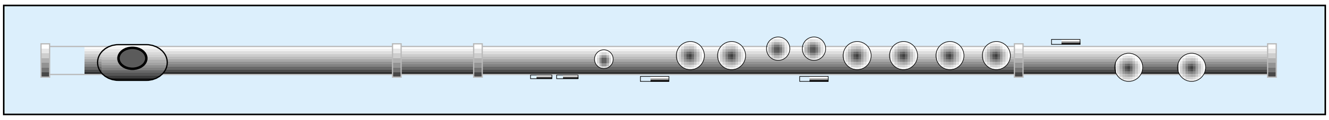
The transverse flute has been known since antiquity, and has been undergoing a constant process of development and improvement ever since. The current form (shown in Figure 3, below) appeared in the 19th century, gaining the keys and extra holes that allow it to produce (almost) exact semitones without half-holing and all the other palaver that we investigated last month. To be honest, it's still not possible to play a perfect chromatic scale across all the flute's registers, but the modern instrument is close to ideal, so a flute patch played from a keyboard is immediately more likely to be realistic than a recorder patch played the same way.

Nonetheless, the size of the holes is still an important factor. Experiments show that larger holes allow the upper partials to 'sound' with greater energy, so the tone is brighter. Earlier, wooden flutes with small holes — generally found in folk music — have a more mellow timbre that can be surprisingly reminiscent of the recorder.

Why surprising? Because, unlike the recorder, the main bore of the flute is almost exactly cylindrical. And, unlike the recorder, most flutes are made of metal rather than wood. Even gold and platinum are suitable materials if the owner is wealthy (and ostentatious) enough. Apparently, instruments made using precious metals sound no better than those drawn from more affordable substances, and even 'affordable' flutes are not exactly cheap, with the best being constructed from an alloy that is 90-percent silver. Cheaper models — those that you would find in a typical high-street music store — are typically drawn from a copper/zinc/nickel alloy covered by a thin layer of silver plating.

▶

Figure 3: The modern orchestral flute, shown without the levers.



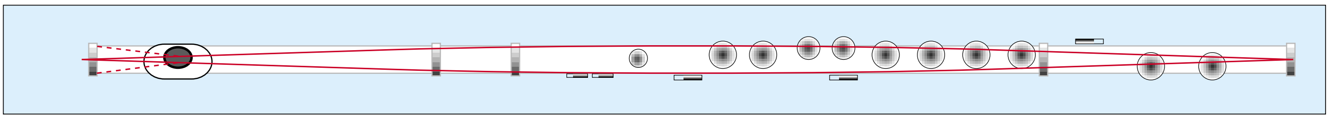


Figure 4(a): The first harmonic (fundamental) of the lowest note playable on a flute.

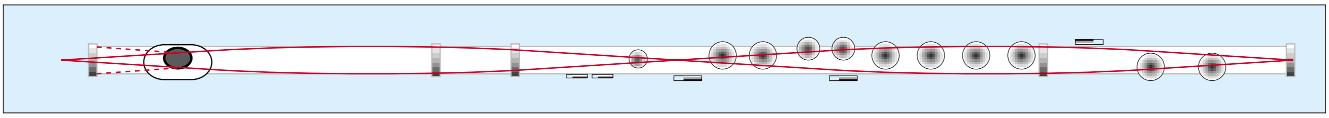


Figure 4(b): The second harmonic of the lowest note playable on a flute.

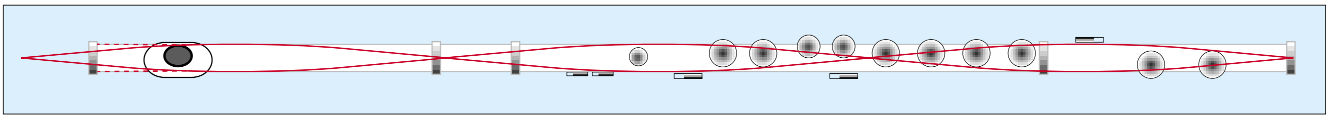


Figure 4(c): The third harmonic of the lowest note playable on a flute.

Far more important than the material from which the flute is constructed are the shape and angle of the embouchure hole, the height of the embouchure chimney, and the position of the cork end-stop that tunes the instrument. A badly adjusted cork can mis-tune the registers by a full semitone relative to one another, and will limit the upper note that the instrument is capable of producing. However, the existence of the cork — or, more accurately, the cavity between the embouchure hole and the cork — has another profound effect, as shown in Figures 4(a), 4(b) and 4(c) above.

As you can see, having the embouchure hole a short distance from the end of the bore causes the flute to act in a different way to a simple pipe, with a pressure maximum at the cork, and a wavelength within the bore that suggests that the effective length of the flute increases for higher harmonics! This is analogous to the 'overshoot' I mentioned when we discussed brass instruments, and its effect is to make the harmonic frequencies more and more approximate with increasing harmonic number. Flute manufacturers try to compensate for this by making tiny adjustments to the position of the cork, the shape of the embouchure chimney, and the sizes and positions of the holes.

As the diagrams show, the modern flute has 16 holes (13 large and three small) which seems a little excessive when all we require is 12 pitches per octave, but none are superfluous and, like the recorder, the flute has many complex fingerings.

As with the German recorder, opening holes progressively from the far end makes the effective length of the instrument shorter, thus raising the pitch a semitone at a time. If you open, say, the lowest four holes on a 'C' flute, you obtain a pitch four semitones higher, which is the 'E' shown in Figure 5. However, this is not the only way to obtain pitches, and just as we found last month, the flute is able to play high pitches with lower holes closed. The most obvious example of this appears in Figure 6 (below), in which the player produces the octave 'C' by opening a single hole almost exactly one half of the way along the pipe. I won't 'bore' you [*Groan — Ed*] with the physics of this, because I think that it is intuitively evident why this should be the case (hint: it makes it impossible for the pipe to support any odd harmonics, including the fundamental). Opening other holes at integer fractions of the bore length causes the instrument to jump to even higher octaves, up to its limit.

Another way in which the flute is similar to the recorder is in the sound produced at low volumes. This is dominated by the fundamental, with rather weak harmonics. But as you blow harder, higher harmonics appear and, as their amplitudes increase, the flute's tone becomes increasingly complex and more sonorous. Strangely, the flute does not get much louder when you blow harder, but does so when you relax your lips to allow a greater cross-section of air to pass.

If you continue to blow harder still, you will eventually 'overblow' the flute and cause its pitch to jump an octave. This is not true of all woodwind instruments, because cylindrical pipes with one closed end have no even harmonics, so overblowing jumps to the third harmonic, one-and-a-half octaves above the fundamental. This has a significant downside: it is not possible to re-use the fingering of one register in another, higher register.

Now, let's consider the sound radiation from a flute. If it were a trumpet, you might expect it to be loudest when the bell is pointing directly as you. However, it is *not* a trumpet, and the radiation properties are mind-bogglingly complex. Even in the simplest case, when all the holes are closed and the flute is acting as a pipe open at both ends, the embouchure hole and the open end are acting as roughly equal amplitude sources. This would be bad enough, but — for perhaps obvious reasons — the odd harmonics of the note are in phase at the

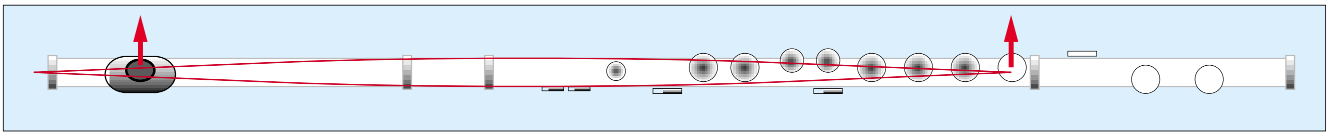
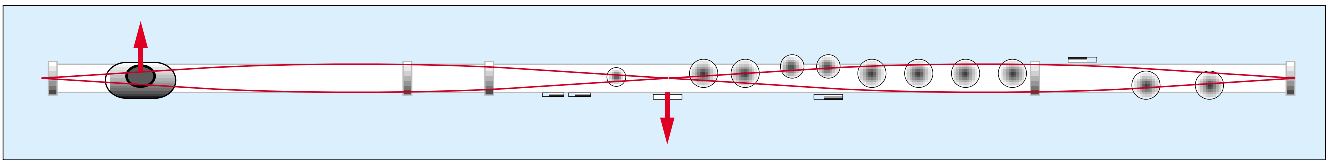


Figure 5: Playing a low 'E' on a 'C' flute.

Figure 6: Playing an octave 'C'.



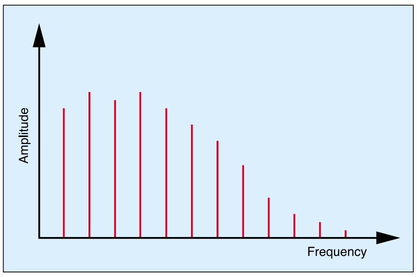


Figure 7: The spectrum of a low ‘C’.

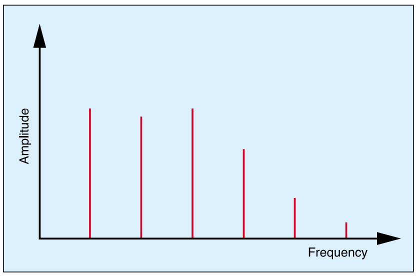


Figure 8: The spectrum of the octave ‘C’.

two ends, whereas the even harmonics are 180 degrees out of phase. This means that there are complex phase interactions between the sources, wherever you listen in the soundfield.

But things get worse! When any of the holes are open, they also radiate sound energy. Given that numerous holes may be open, each radiating at different phases, the tone at different points in the soundfield can differ considerably. Fortunately, you will usually hear a flute played in a reverberant space in which the various cancellations and reinforcements cancel out to give a smoother spectrum. This explains why the use of a good reverb unit is so critical when synthesizing a convincing flute sound. Which brings us neatly to...

## Synthesizing The Flute

Although it may seem that we've just investigated the flute in some detail, we have barely scratched the surface of its complexities. Seemingly trivial things such as the porous properties and rigidity of the inner surface of the bore, the smoothness of the bore, the height of the chimneys beneath the holes, the thickness of the pipe at various places along the instrument... all these and more affect the timbre. Of course, there's no space to investigate these here, and even if we did, it would take us into areas of acoustics and mathematics that are best avoided. So let's ignore the flute's secondary characteristics, and concentrate on its primary attributes. We'll start by looking at its harmonic spectrum, and the way that the sound changes in time.

Figure 7 (above) shows the spectrum of a low note such as the bottom ‘C’. Played with moderate force, it is rich in harmonics, with strong contributions from the second, third and fourth harmonics, followed by a reasonably ‘1/n’-like shape from the fourth upward, until the spectrum disappears at some upper frequency. The ‘C’ played with the same force an octave higher demonstrates a similar pattern, with a suppressed fundamental and second harmonic, but the ‘1/n’ shape for the third harmonic up to the same upper frequency (see Figure 8, above). If we jump an octave higher, we find that a ‘1/n’ shape is demonstrated for all harmonics, but that there are very few of these, because the upper frequency limit still applies.

You might wonder why the harmonic spectrum is so truncated, but the reason is clear: As the frequency of the oscillation in the pipe increases, it strays further and further from the true resonant frequencies supported by the pipe, and eventually the two are far enough apart to ensure that no standing wave is supported. This is the consequence of the behaviour shown in Figures 4(a), 4(b) and 4(c). In the standard orchestral flute, the cutoff frequency is 2kHz or thereabouts, which explains why all the spectra in Figures 7, 8 and 9 are truncated at this frequency, no matter what note is played.

You might think that this is not something that we can synthesize without recourse to additive synthesis, but fear not... there is a simple way to achieve all three of these spectra using three analogue modules: a sawtooth oscillator, and voltage-controlled low-pass and high-pass filters.

Consider Figure 10 (above), and you can see how the fundamental and low harmonics can be suppressed for a low note but not for a higher one, and how a similar, if not identical upper cutoff frequency can be imposed on all notes, regardless of pitch. Experiments show that, depending upon the filter slope, the cutoff frequency of the high-pass filter should be in the region of a few hundred Hertz. The cutoff frequency of the low-pass filter resides — as discussed — at 2kHz, although I have found that pitch tracking of a few percent is necessary to ensure that high notes are reproduced with the correct brightness relative to low notes.

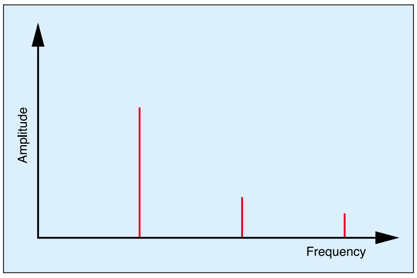


Figure 9: The spectrum of the ‘C’ in the next register.

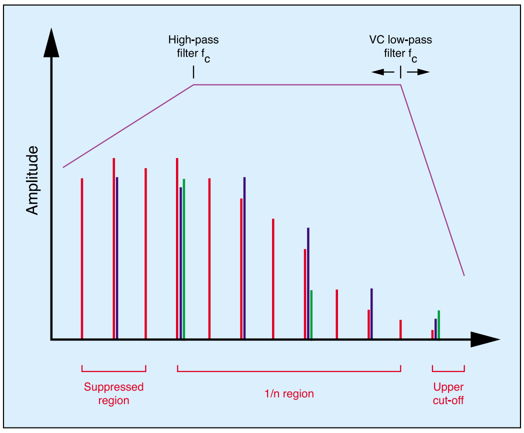


Figure 10: Creating the flute's harmonic spectrum.

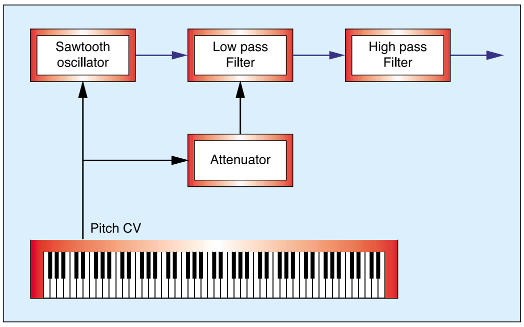


Figure 11: Creating the spectra in Figures 7, 8 and 9.

The requisite architecture is shown in Figure 11 (below), and is common to many analogue monosynths. This is an elegant result, although it is complicated slightly if we take into account notes played with different blowing pressures. Measurements show that the fundamentals and low harmonics of softly played notes are just as loud as forcefully played notes, but that the upper harmonics are attenuated to a greater extent. This means that, with a pressure-sensitive synth, we need only route aftertouch to the low-pass filter cutoff frequency to achieve a realistic result, as shown in Figure 12 (on the next page) This is, of course, what the ARP Pro Soloist offers, and one of the reasons why its sound is so satisfying... Press harder, and the sound becomes brighter. Press less hard, and the sound becomes less bright. The equivalence to blowing pressure could not be closer. By the way, the spectra in Figures 7 to 9 are idealised representations that

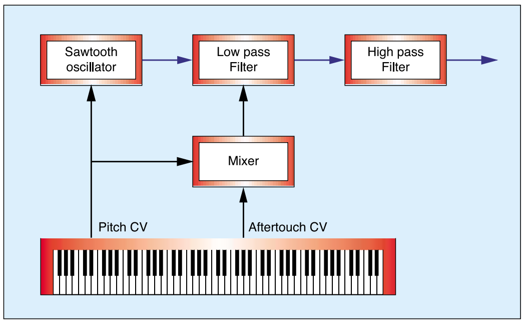


Figure 12: Imitating changes in blowing pressure.

make no mention of the noise that is inherent in the flute's sound. Unfortunately, we cannot synthesize this accurately without recourse to large synths. Some programmers add a little white noise to the primary oscillator signal, but personally, I think that this is a mistake, detracting from the authenticity of the patch rather than adding to it.

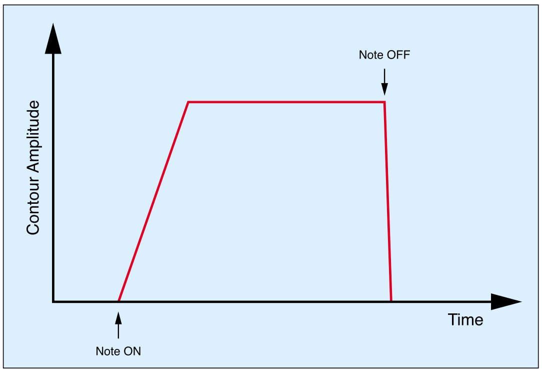


Figure 13: The loudness contour of a flute played without a 'chiff'.

Now let's determine the amplitude contour of the sound. Because the flute is to some extent 'on' or 'off', this proves to be simple. To a first approximation, the Attack must not be instantaneous, but neither must it be too slow, there need be no Decay stage, the Sustain must be at maximum, and the Release is quick (see Figure 13, above). However, this proves to be less than ideal, because the synth will reinitiate its envelope from zero whenever a fresh note is played, resulting in the unnatural 'sucking' sound represented by (although exaggerated in) Figure 14 (see right).

We compensate for this by increasing the Release time (see Figure 15) to the point where it overcomes this (see Figure 16) without adding an inappropriate, slow release to the sound. When set correctly, this gives a nice, natural articulation between the notes in a legato passage, while still allowing you to play in a moderately staccato fashion if desired.

Although I've already discussed the next point, the following sentence is the key to synthesizing the sound of the flute, so I'm going to make a big issue of it:

*A flute does not get significantly louder or softer as the player alters the blowing pressure; it becomes brighter or duller.*

This means that we can use the same contour for the low-pass filter as for the amplifier. Furthermore, unlike almost all of the instruments we have studied so far, the flute does not exhibit pitch modulation (vibrato) or even loudness modulation (tremolo) but rather tonal (brightness) modulation. Tests show that trained players tend to vary the blowing pressure by about ±10 percent with a frequency of around 5Hz to 6Hz, and we can synthesize this easily using the simple patch in Figure 17 (on the next page).

Adding pressure-sensitivity is also easy, as demonstrated by Figure 18, in which I have routed the pressure CV to both the overall brightness and to the amount of modulation. If your synth is able to respond in this fashion, it will add immeasurably to the expressiveness of the sound. Nevertheless, the expanded patch remains much simpler — and therefore cheaper — than the huge patch in Figure 1.

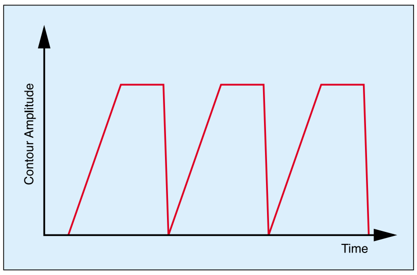


Figure 14: This sucks!

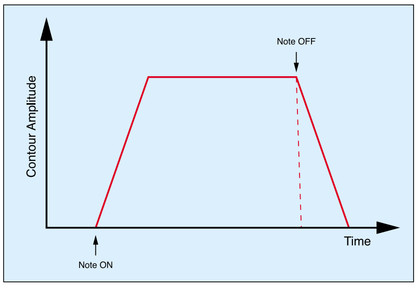


Figure 15: Extending the Release time..

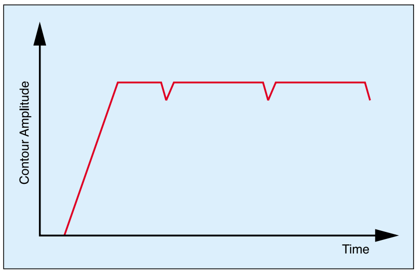
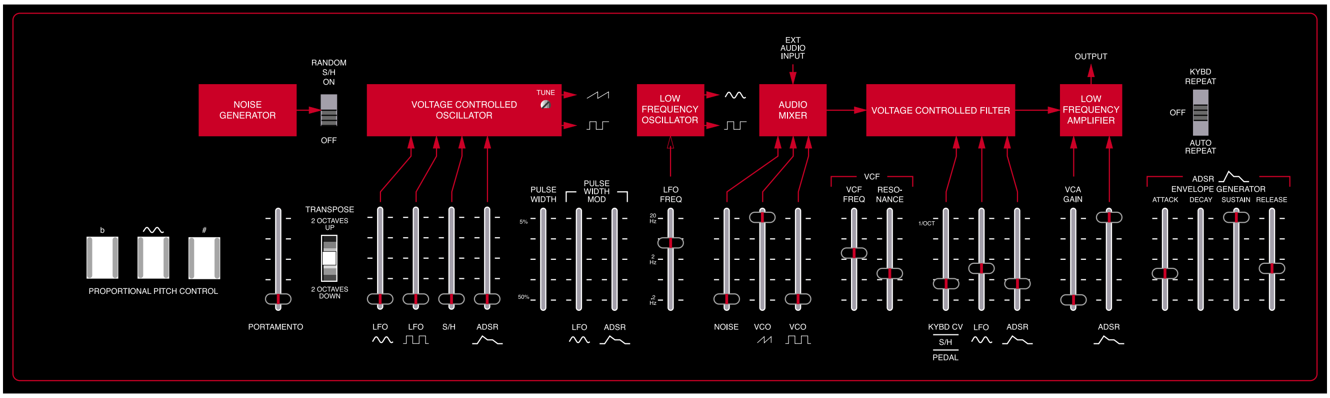


Figure 16: This does not suck!

Returning to the 'insensitive' patch in Figure 17, there's only one family of synths that reproduces the basic flute sound as well as I would like. The Minimoog won't do it; neither will the Roland SH101. The Yamaha CS-series and the Korg MS10 and MS20 also fall short of the mark, although the Minikorg 700S is capable of a remarkable flute patch.

Nonetheless, I always return to ARP synths for my analogue flute patches. Other than the pressure-sensitive Pro Soloist, the Odyssey is my instrument of choice, not

Figure 19: The Axxe flute.



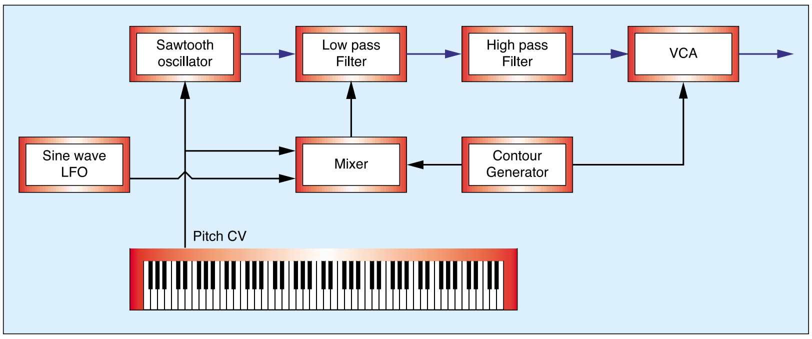


Figure 17: A simple flute patch.

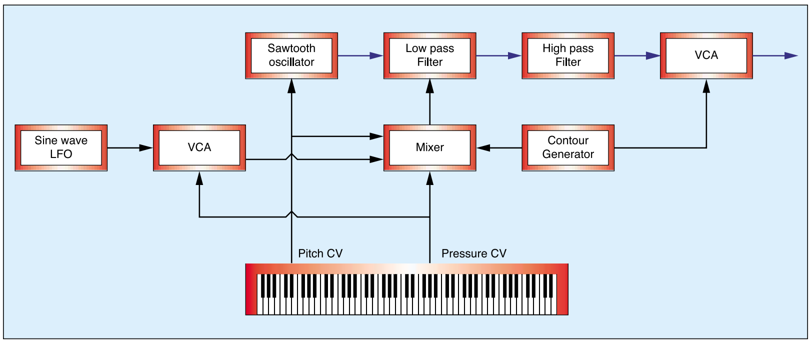


Figure 18: Adding pressure sensitivity.

because it has two oscillators and all manner of other sophisticated gubbins, but because it has a high-pass filter. However, even the lowly Axxe is capable of a superb flute, as I will now show.

## The Axxe Flute

We'll start by determining the pitch and pitch modulation for each note. As you can see in Figure 19 (below left), nothing other than the keyboard CV affects the oscillator pitch. And, because there is no pulse wave in the sound, the positions of the PWM controls are irrelevant.

The same diagram shows that the LFO frequency is set in the correct range of 5-6Hz, and is applied to the low-pass filter cutoff frequency, together with a little pitch CV tracking and a smidgen of ADSR contour. Delayed modulation would be even better here, helping to imitate the way in which a flautist would introduce the movement in the sound after the initial chiff, but the Axxe has no way to produce this.

The filter itself is set to exactly 2kHz using the tuning trick I described in Synth Secrets 49 (see *SOS* May 2003, or surf to: www.soundonsound.com/sos/may03/articles/synthsecrets49.asp). You'll also notice that I've applied a moderate amount of filter resonance. Used carefully, this acts a little like a high-pass filter, adding an edge to the sound without making it sound electronic or synthetic.

Looking further to the right of Figure 19, you can see how the patch is completed by the application of the appropriate contour to the audio signal VCA. If you don't have access to an Axxe to try this, you'll just have to accept my word that the results are more than acceptable. This, then, begs the question, "Why do ARP synths get this sound right in a way that many other synths do not?" The answer is the same as I've offered before when I've raised this question: not all synth oscillators, filters and signal paths are created equal, or even equivalent. The ARP 4034 filter used in the Pro Soloist and Explorer, and the 4075 found in the Axxe and Solus happen to excel at brass and flutes, although there are many sounds for which they are not the devices of choice. That's why it's so important to have access to a range of electronic instruments whenever possible.

## Epilogue

In the past three months, we've learned much about recorders and flutes, such as the effects of turbulence in pan flutes, of adding holes to create shakuhachis and recorders, and the difference between vibrato, tremolo, and the filter modulation that is so important this month. With knowledge like this, you'll be able to program all manner of realistic and not-so-realistic flute-like sounds, instead of stepping through 1000 presets to find a sound that merely approximates what you want. SOS

# Synth Secrets

## Synthesizing Tonewheel Organs

Gordon Reid

**Long before Bob Moog built his first synth, there was the Hammond tonewheel organ; effectively an additive synthesizer, albeit electromechanical rather than electronic. So emulating a Hammond with an analogue synth shouldn't be too hard, right? Well...**

Long before Keith Emerson and Rick Wakeman showed us that keyboard players did not have to be accompanists dressed in black and illuminated by black spotlights, and even longer before musicians began to take to the stage armed with nothing but a laptop computer and a pair of turntables, jazz and blues organists were the hi-tech musicians of their day. So when players such as Jimmy Smith and Earl Grant cast off their sackcloth and made a bee-line for the front of the stage, they did so with nary a Minimoog, ARP 2600, EMS VCS3, chorus unit, phaser, ensemble, or digital reverb in sight — which isn't surprising, as none of these had yet been invented. With no more than a Hammond organ, a bit of spring reverb, and maybe a touch of overdrive, these guys were creating exciting new forms of dance music throughout the middle of the 20th century. In retrospect, it's far from unreasonable to suggest that almost all modern forms of hi-tech music evolved from the 'black' music of the 1940s and 1950s, and it is therefore appropriate to hand the award for most influential keyboard instrument of the 20th Century to the Hammond 'tonewheel' organ.

### A Course In Electromechanics

Like many brilliant ideas, the basis of Laurens Hammond's tonewheel generator is simple: a knobbly wheel rotates in the presence of a magnet, and the resulting changes in the magnetic field induce a signal in a pickup (see Figure 1, below).

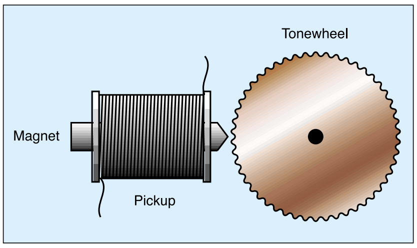


Figure 1: A single Hammond 'tonewheel' and pickup.

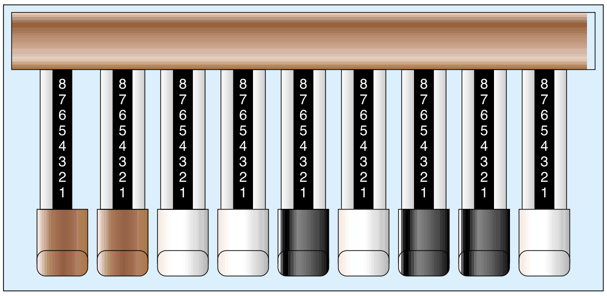


Figure 2: The nine 'drawbars' fully extended.

The waveform and frequency of the signal is determined by the shape of the wheel and the number of 'bumps' that pass the tip of the magnet every second. Given that in the finished instrument, all the tonewheels are mounted on a single axle, different frequencies are obtained not by using different rotation speeds, but by using tonewheels of different sizes and geometries. Like I said... brilliant!

When designing his organ, Hammond decided that each tonewheel should generate a sound as close as possible to a sine wave, so that players could construct timbres using a fundamental and overtones. Building on this idea, he chose a system by which players could mix up to nine sine waves simultaneously, using 'drawbars' (see Figure 2) to give each an amplitude ranging from zero to eight. Some later Hammonds offered more drawbars, and some offered fewer, but nine is the classic configuration.

The lowest pitch on a full console Hammond is 16', with drawbars at five and two-thirds feet (5 2/3'), 8', 4', two and two-thirds feet (2 2/3'), 2', one and three-fifths feet (1 3/5'), one and one-third feet (1 1/3'), and 1'. So, despite Hammond's strange decision to call the 8' the fundamental (or 'Unison') and the 16' drawbar the sub-octave, the 16' pitch is the fundamental of a series that includes the first, second, third, fourth, sixth, eighth, 10th, 12th and 16th harmonics, as shown in the table below.

Different drawbar configurations are called 'registrations', and (if my maths is correct) there are 387,420,489 of these on each manual. These registrations fall into

| DRAWBAR | COLOUR | PITCH | TRADITIONAL NAME | HARMONIC NUMBER |
|---|---|---|---|---|
| 16' | Brown | Sub-octave | Bass | 1 |
| 5 2/3' | Brown | 5th | Quint | 3 |
| 8' | White | Unison | Neutral | 2 |
| 4' | White | 8th | Octave | 4 |
| 2 2/3' | Black | 12th | Nazard | 6 |
| 2' | White | 15th | Block-flöte | 8 |
| 1 3/5' | Black | 17th | Tierce | 10 |
| 1 1/3' | Black | 19th | Larigot | 12 |
| 1' | White | 22nd | Sifflöte | 16 |

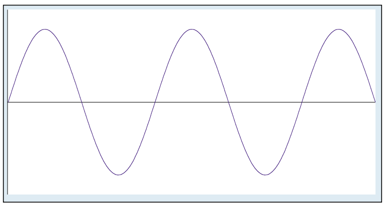


Figure 3: Hammond registration 80 0000 000.

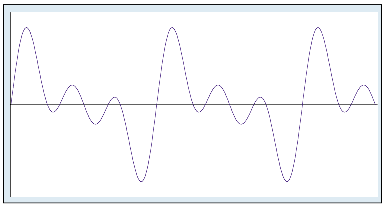


Figure 4: Hammond registration 88 8000 000.

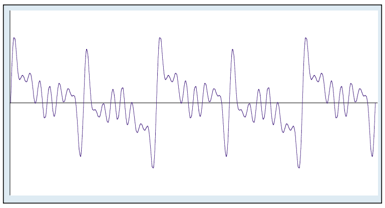


Figure 5: Hammond registration 88 8888 888.

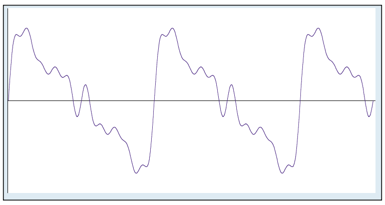


Figure 6: Hammond registration 83 4211 100 (slightly sawtooth-ish?).

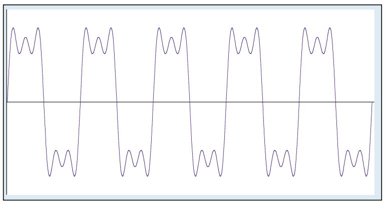


Figure 7: Hammond registration 00 8030 200 (slightly square-ish?).

groups with archaic names such as 'Stopped Flutes', 'Half-covered Flutes', 'Gemshorns', 'Strings', 'Vox Humanae', 'Reeds'... and so on. Within each of these there are anywhere between a few hundred and a few million unique combinations, and each can be represented by a nine-digit number written in the form 'xx xxxx xxx'. So, for example, if the 16' drawbar is fully extended but all the others are pushed home, we can write the resulting registration as 80 0000 000.

Now, if each drawbar produces a sine wave, 80 0000 000 will not create a very interesting sound. Depending upon the amount by which you pull out the 16' drawbar, you will simply obtain a sine wave of greater or lesser amplitude (see Figure 3, left). So you add interest by pulling out combinations of drawbars to create complex registrations. Figure 4 shows the waveform generated by one of the simplest but most important of these, beloved of Jimmy Smith, Keith Emerson, and heavy rock players the world over. The registration is 88 8000 000 and, if you are an Hammond *aficionado*, you will immediately recognise its punchy timbre.

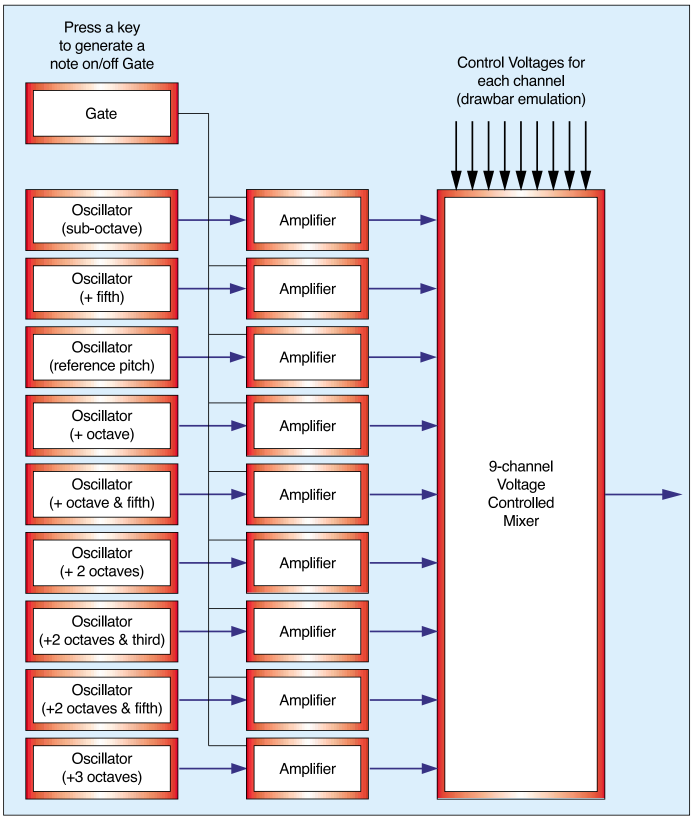
In contrast to the simplicity of 88 8000 000, and often deprecated by classical organists, is the registration 88 8888 888 (see Figure 5). This has all nine harmonics present at maximum amplitude, and is very full and bright.

More interesting, perhaps, are the registrations shown in Figures 6 and 7. The first of these is 83 4211 100, the closest approximation available to a '1/n' harmonic series, while the second is 00 8030 200, an approximation to a '1/n' series with all the even harmonics missing. In other words, they are the closest a vintage Hammond can come to producing a sawtooth wave and a square wave, respectively.

Clearly, we can create a huge range of tones using the nine pitches available and, way back in the mists of time (well... in part 14 of this series, to be precise — see *SOS* June 2000, or surf to www.soundonsound.com/sos/jun00/articles/synthsec.htm), I showed how we can use nine sine-wave oscillators, nine amplifiers, a gate of some sort and a mixer to emulate a note produced by a tonewheel generator. Figure 8 (on the next page) shows an advanced version of this idea, with the oscillators' pitches fixed to the drawbars' pitch relationships, and a voltage-controlled mixer that allows you to mix the oscillators' outputs just as you would if you were clutching a fistful of drawbars.

Apart from dedicated additive instruments such as the Kurzweil 150, Kawai K5 and Kawai K5000 (of which more next month), there is only one family of synths that allows you to patch Figure 8 in a cost-effective fashion. These are the more powerful of the FM synths that dominated the mid- to late-1980s. The DX7 isn't quite up to the job, but the DX5 and DX1 have a dozen freely tuneable 'operators' so, using Algorithm 32, you can program Figure 8 with three oscillators to spare. Long before the current crop of digital B3/C3 emulators, these powerful synths were responsible for some excellent

Figure 8: You need 20 modules for each note of an additive Hammond emulator!


Press a key to generate a note on/off Gate
Gate
Control Voltages for each channel (drawbar emulation)
Oscillator (sub-octave)
Oscillator (+ fifth)
Oscillator (reference pitch)
Oscillator (+ octave)
Oscillator (+ octave & fifth)
Oscillator (+ 2 octaves)
Oscillator (+2 octaves & third)
Oscillator (+2 octaves & fifth)
Oscillator (+3 octaves)
Amplifier
Amplifier
Amplifier
Amplifier
Amplifier
Amplifier
Amplifier
Amplifier
Amplifier
9-channel Voltage Controlled Mixer

Hammond impersonations.

Unfortunately, there are few of the larger DXs in circulation, and you're unlikely to lay your hands on one. If you do, you'll probably pay up to £750 for a DX5, and as much as £2000 for a DX1. Oh, alright... I admit that this is *not* a very cost-effective solution! So let's see whether we can use a much simpler and cheaper analogue polysynth to patch an acceptable Hammond sound.

## Organ-ism

Back in the dim and distant 1980s, I owned two Hammond emulators: a cheap and cheerful Crumar Organiser that had cost me the grand sum of £199 in the late '70s, and a Korg BX3 that, a few years later, had cost a whole lot more. The Crumar sounded little like a Hammond, but was relatively light and portable. In contrast, the Korg sounded far more realistic, but was almost as unwieldy as the top of a split B3. As a result, I was always looking for alternatives that would sound good, but save weight and hassle.

I tried everything, but — until the advent of digital emulators such as the Hammond XB2 several years later, I found that nothing improved greatly upon the 88 8000 000 organ sound that I patched on a very simple analogue polysynth. That synth was a Roland Juno 6, and given that it offered just one oscillator per voice and no sophisticated voicing capabilities, it seemed a most unlikely solution to my problem.

I'll start to develop the patch by considering the Juno's single oscillator section (see Figure 9). As you can see, this offers just two waveforms — variable pulse (with pulse-width modulation) and sawtooth — plus a square-wave sub-oscillator one octave below the basic pitch. There is no way to mix the pulse and sawtooth waveforms in different amounts — they are either 'on' or 'off', although you can add as much or as little sub-oscillator as you like.

I'm now going to introduce a rather unusual way to represent harmonic spectra. I haven't used this representation before, but it's particularly well-suited to depicting the output from tonewheel organs.

For reasons that will soon become apparent, I will draw the harmonics' frequencies and amplitudes on logarithmic scales. I will also invert the amplitude axis so that the louder a harmonic is, the *lower* on the page it appears. Strange though this may seem, it mimics a visual representation of Hammond drawbars. So, for example, I can depict a spectrum comprising three sine waves lying on the first three harmonics of a given frequency (see Figure 10, below), and it is should be clear that this is a different way of representing the 88 8000 000 registration shown in Figure 11).

If we now return to the Juno 6 and activate its sub-oscillator, we will (in a perfect world) obtain the spectrum shown in Figure 12 (on the next page). Clearly, this is a million miles from what we require. What's more, for a fundamental of, say, 200Hz, there are 100 harmonics within the

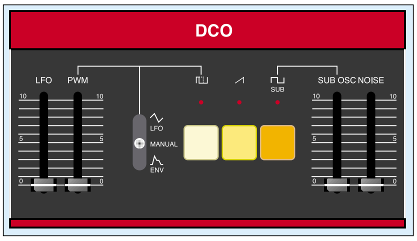


Figure 9: The Juno 6 Digitally Controlled Oscillator.

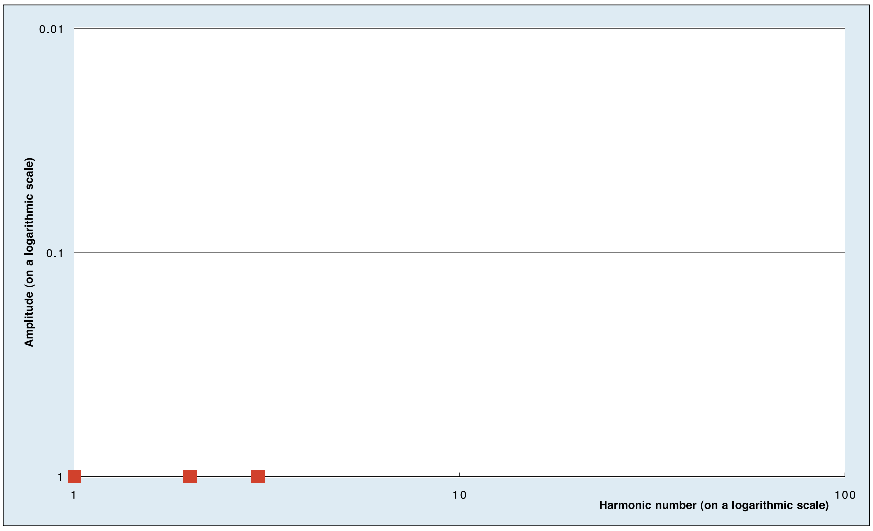


Figure 10: Representing a sound with three harmonics (the first, second and third) of equal amplitude.

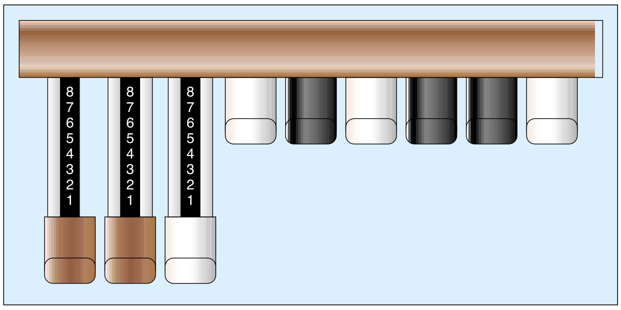


Figure 11: The sound represented in Figure 10, created using 16', 5 2/3' and 8' drawbars.

▶ 20Hz-20kHz audio spectrum, of which 50 have non-zero amplitude. By the way, I hope that you can now see why it's useful to plot '1/n' plots on logarithmic axes — on linear axes, this graph would have been considerably wider than this magazine, and the resulting fold-out diagram would have given *SOS*'s printers a terrible headache!

To start sculpting this into something useful, I am going to filter the sub-oscillator using the Juno's low-pass filter, with the cutoff frequency set precisely 19 semitones (one octave and a fifth) above the sub-oscillator pitch itself. The reasons for this very precise setting of the cutoff will become apparent shortly...

The result appears in Figure 13 (right). As you can see, the first two partials (which are the first and third harmonics) pass through the filter unscathed, while the third (the fifth harmonic) is attenuated, and the higher harmonics are so quiet as to be almost inaudible. This is closer to Figure 10, but still wins no cookies.

Now I'm going to add the output from the oscillator. I'll set it up so that only the pulse wave is produced, and this has a duty cycle of 'one third'. If you recall the instalment of this series in *SOS* March 2003 (also located at www.soundonsound.com/sos/mar03/articles/synthsecrets47.asp), and specifically the large box in that instalment on the nature of pulse waveforms, you'll remember that you can approximate the harmonic content of the resulting waveform if you take a sawtooth wave and remove every third harmonic. Of course, if you remember the rest of that instalment, you'll also recall that this approximation breaks down as you decrease the duty cycle — so the harmonic content of a pulse wave with a duty cycle of one-twelfth, for example, *isn't* much like that of a sawtooth with every 12th harmonic removed at all. But for a pulse wave with a duty cycle of one third, the approximation is reasonably sound, and the remaining partials conform almost exactly to a 1/n amplitude spectrum, as demonstrated in Figure 14 (right).

The output from the pulse wave has to pass through the same filter as the sub-oscillator, so it too will be heavily filtered. However, whereas the filter cutoff frequency is set to the third harmonic of the sub-oscillator, it lies halfway between the fundamental of the pulse wave and its first overtone (which, in this case, is the second harmonic). So — to paraphrase the above — the fundamental passes through the filter unscathed, but the first overtone is attenuated and everything else is so quiet as to be almost inaudible (see Figure 15 on the next page).

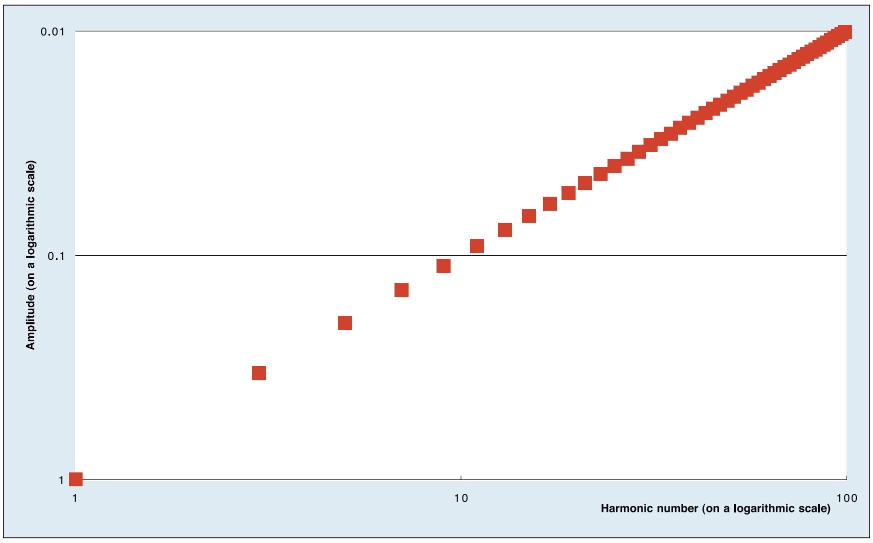


Figure 12: The first 50 harmonics of a mathematically perfect square wave, shown on logarithmic axes.

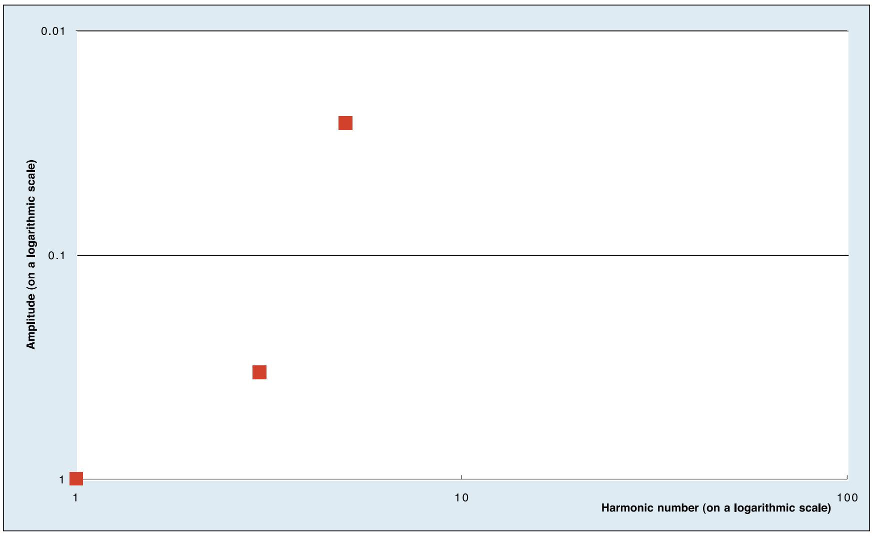


Figure 13: Filtering the sub-oscillator from the third harmonic upwards.

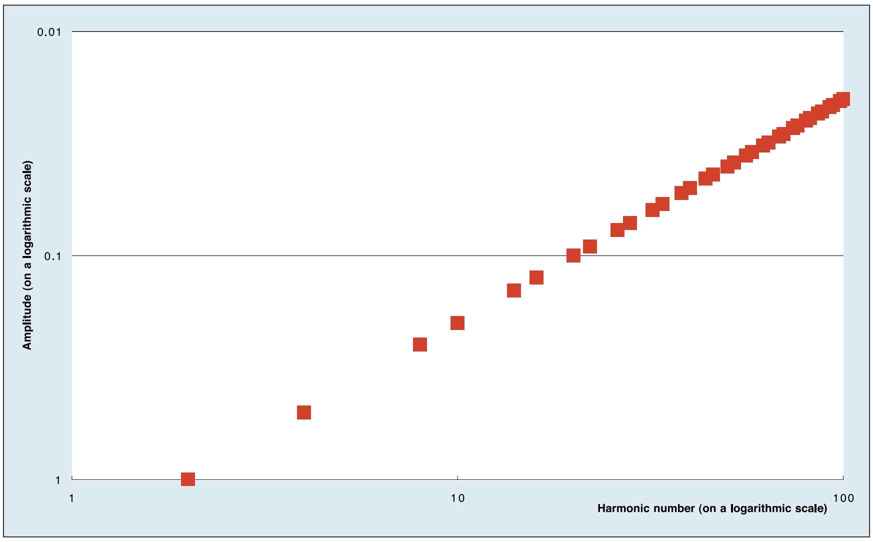


Figure 14: The spectrum of a 33-percent pulse wave.

I'll now switch on the Juno's pulse wave and sub-oscillator simultaneously, and show the spectrum of the mixed signal by adding the partials in Figures 13 and 15. The result of this can be represented in Figure 16, also on the next page). Clearly, this is much ▶

closer to the ideal, with the leftmost partials representing the 16’ and 8’ drawbars fully extended, and the next two representing the 5 2/3’ and 4’ drawbars respectively. The only aberration is the fifth partial, which has an amplitude of about 2.5 percent.

As shown by the table earlier in this article, there *is* no Hammond drawbar which produces the fifth harmonic, so in theory this should not be present. But in the real world, impurities in the geometry of the tonewheels and valve distortion add some fifth harmonic to the sound, so this does not overly concern me.

Nonetheless, it would be nice to bring the 5 2/3’ pitch to the fore because, as it stands, the sound lacks depth (if you pull out just the 16’ and 8’ drawbars on a Hammond, you obtain a relatively uninteresting timbre, so it’s not surprising that the synthesized equivalent should be similarly lacking).

Before attempting to raise the profile of the third harmonic in this way, let’s check the settings for the DCO, as shown in Figure 17. Note that the sawtooth wave is ‘off’, that the pulse wave modulation switch is set to ‘Man’ (manual), and that the PWM slider is positioned so that the pulse width is a constant 33.33 percent. With practice, you can adjust this by ear... as you move the slider to the correct position, you can hear the third harmonic disappear. Note also that the sub-oscillator output is at its full amplitude, but that there is no contribution from the noise generator.

Now it’s time to return to that troublesome 5 2/3’ drawbar. Given that we have no further control over the oscillator, how can we accentuate the third harmonic of the sub-oscillator?

The secret lies in the filter which — if you remember — is tuned exactly 19 semitones above the sub-oscillator’s pitch. And of course, 19 semitones above the sub-oscillator is where the third harmonic lies...

If you’ve wondering how on Earth this helps, it should help to know that the Juno 6 has a self-oscillating filter that tracks the keyboard perfectly. If we set the filter resonance to 100 percent, the self-oscillating filter produces a sine wave at the filter cutoff frequency — in other words, 19 semitones above the fundamental. So, if the sub-oscillator produces a bottom ‘C’ and the pulse wave produces the ‘C’ an octave higher, the self-oscillating filter will produce a ‘G’ 1.5 octaves above the sub-oscillator.

This works on the Juno because its filter is so perfectly behaved. Unfortunately, attempting this trick on most other analogue synths causes all manner of problems, including severe attenuation of the lower frequencies, and unpleasant distortion as the signal presented to the filter input ‘fights’ the signal generated within the filter. Oh yes... and it’s unlikely that the sine wave produced by self-oscillation will track the keyboard correctly, so its pitch will wander all over the place, making the patch useless. So, if you’re trying to create this sound on a lesser instrument (and that includes all the Prophets, all the Oberheim OB-series, and nearly everything else) you must reduce the resonance, leaving it high enough to *amplify*

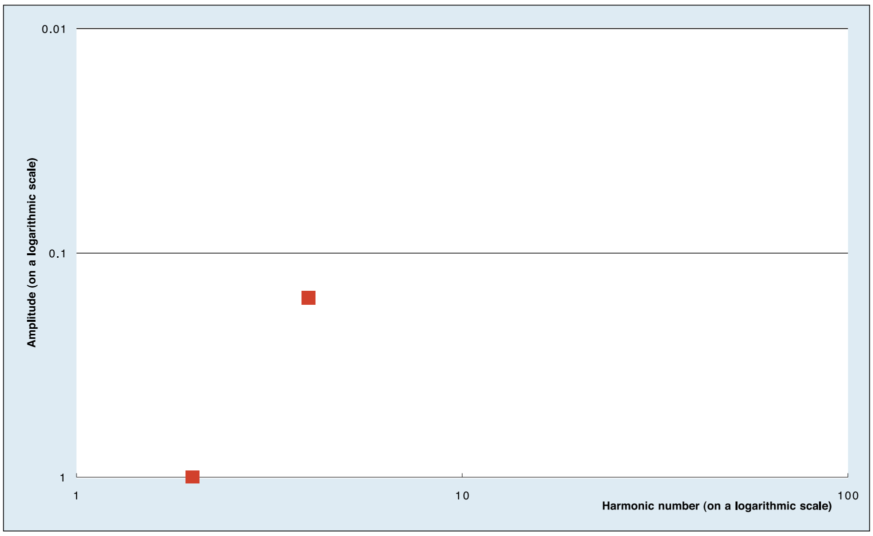


Figure 15: The spectrum of the filtered 33-percent pulse wave.

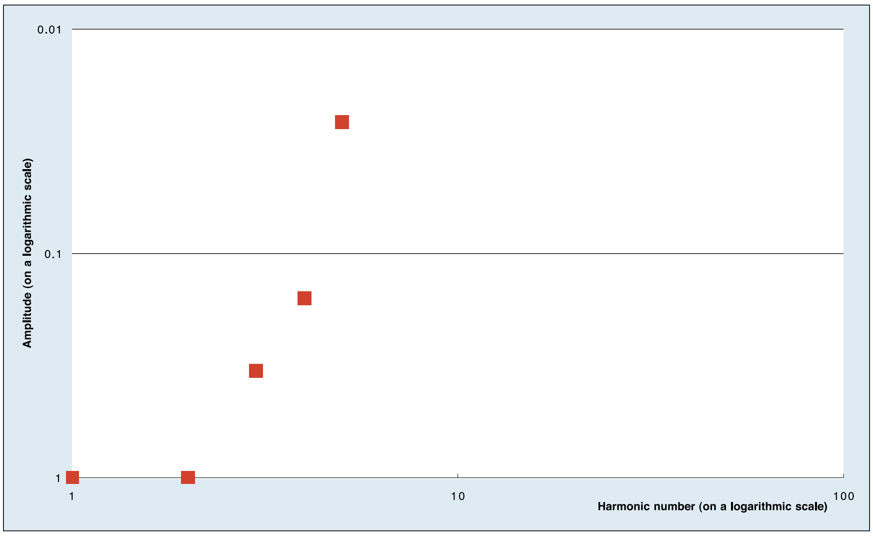
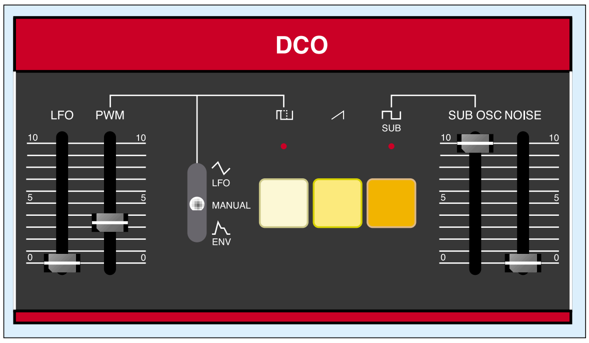


Figure 16: Adding the filtered sub-oscillator and 33-percent pulse wave.

Figure 17: The Juno 6 DCO set to produce a Hammond sound.



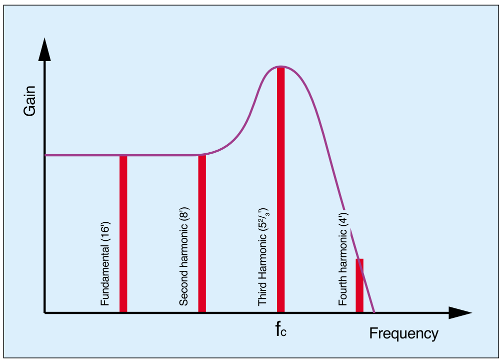


Figure 18: Amplifying the third harmonic using filter resonance.

the third harmonic that is already present in Figure 16, but not so high as to send the filter into oscillation (see Figure 18, on the previous page).

Nice though the result in Figure 18 is (especially if high resonance causes the filter to attenuate the higher harmonics further), I don't see why I should limit myself in this fashion. So I'm going to push the Juno 6's filter all the way into self-oscillation (as shown in Figure 19), creating a pure tone at the 19th semitone, and at the same time severely attenuating all the frequencies that lie above this. The result appears in Figure 20 (below), and is exactly what we were after in Figure 10 — a very elegant result, if I say so myself!

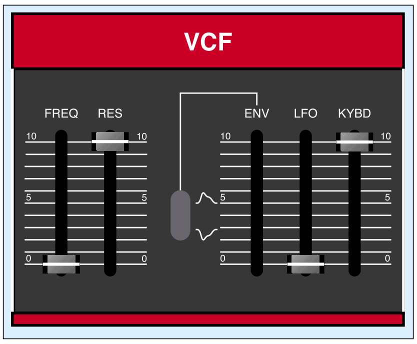


Figure 19: Four of the six Juno 6 VCF settings.

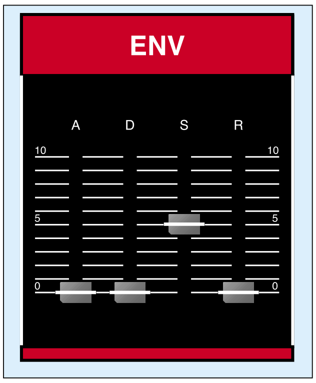


Figure 21: The Juno 6 'Env' settings.

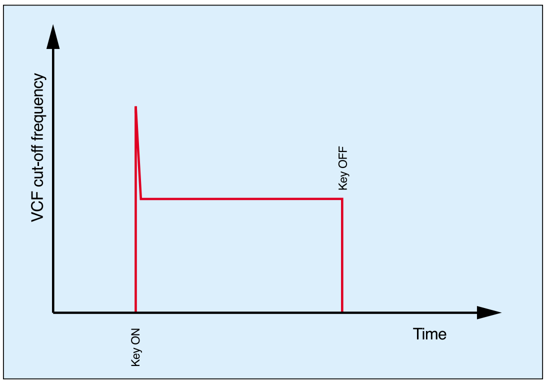


Figure 22: The resulting VCF contour.

However, we need to set up the rest of the filter correctly if the sound is to work. In particular, precise adjustment of the filter envelope settings (which I have omitted from Figure 20) is vital if the cutoff frequency is to lie at the correct pitch. But why do we need to modulate the filter using the envelope? Surely it would be best to leave well alone?

We all know that Hammond organs exhibit a 'spit' at the start of the note, caused by what Laurens Hammond thought were deficiencies in the keying system. Today, of course, we are rather attached to these so-called 'key-clicks', and the programmers of DSP-driven Hammond emulators spend a great deal of time imitating them as accurately as possible. Unfortunately, the Juno 6 lacks the sophistication needed to produce the clicks correctly, so I will have to resort to using the filter envelope to generate a reasonable imitation.

Figures 21 and 22 (above) show how we set up the ADSR envelope generator to create a pronounced, but almost instantaneous, transient at the moment you press a key, and how this can make the VCF cutoff frequency change as you play a note. Given that the filter is oscillating, this will create a very rapid downward sweep during the Decay stage, also accentuating the pulse wave's and sub-oscillator's harmonics as it does so. The 'blip' thus produced is satisfactory for our purposes.

The settings in Figure 21 may look trivial, but to apply the contour to the filter itself, you must position the 'VCF Env' switch for positive polarity and raise the 'Env' slider in the VCF section — see Figure 23 (below). You must then be very careful how you set this up, because the Sustain Level and the amount of 'Env' will together *raise* the cutoff frequency that you have previously tuned so carefully to the sub-oscillator's third harmonic, thus destroying all your hard work so far. So... how can you create the 'key-click' and still get the filter to produce the sound of the 5 2/3' drawbar?

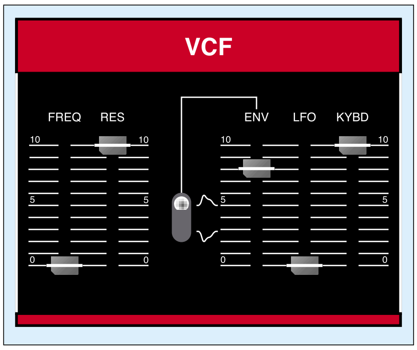


Figure 23: Raising 'Env' amount to apply the ADSR to the filter cutoff frequency.

You solve this conundrum by taking the following steps:

- Set the Sustain Level in the ADSR so that you obtain the amount of 'spit' required. A high value will reduce the amount, while a low value will accentuate it, making the organ very 'clicky'.
- Add the correct amount of 'Env' to the filter to create the click effect that you want.
- Re-adjust the filter cutoff frequency ('Freq') so that the combined effects of 'Freq', 'Env' and Sustain again tune the cutoff frequency to the third harmonic.

The chances are that you'll have to run through these steps a couple of times before everything is hunky dory, but it's not hard

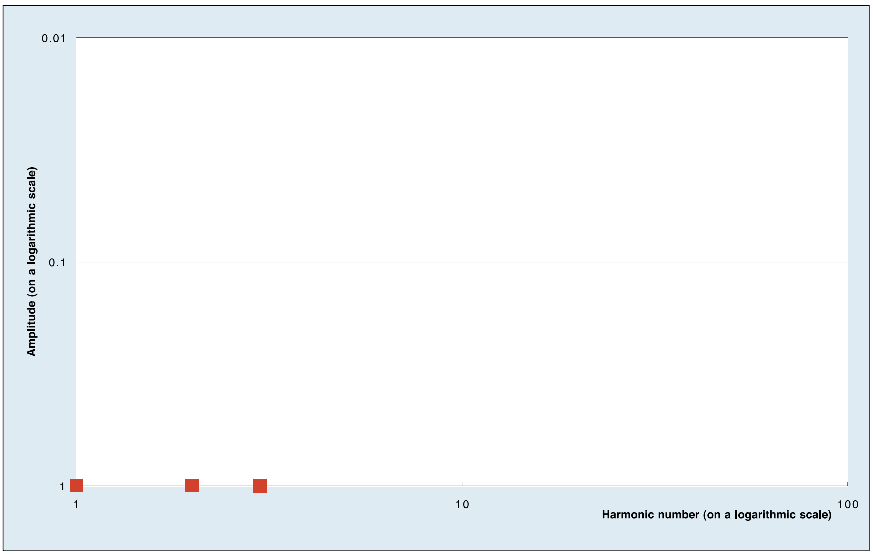
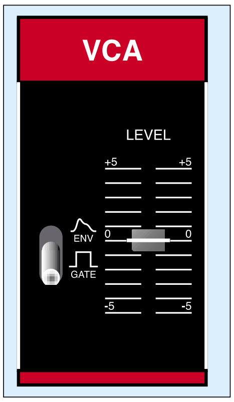


Figure 20: The harmonic amplitudes of the signal after programming maximum resonance in the Juno's filter.

once you've got the hang of it. Personally, I find that a 'Freq' value of 'zero' is best, and that tuning the filter using the 'Env' control alone is the ideal solution.

Now let's take care of the amplitude envelope. To a first approximation, this is rectangular: you press a key and the note immediately attains its maximum amplitude; you release the key and it immediately falls to silence. The Juno 6 has a neat way of achieving this; you can disconnect the VCA from the envelope generator using the switch shown in Figure 24 (right). The amplifier then responds to the gate pulse itself, being 'on' when you press a key, and 'off' when you release it. This is the mechanism we were after way back in Figure 8, and it produces the amplitude contour shown in Figure 25 (below).

Figure 24: The Juno 6 VCA settings.



## Putting It All Together

We now have everything in place to allow us to emulate the tonewheel generator set to an 88 8000 000 registration, so let's combine the parts to create our final synthesized Hammond patch. Figure 26 (at the bottom of the page) does this. Note that the Key Transpose and Hold buttons are off, that the arpeggiator is off, and that there is no LFO applied in either the DCO section or the filter section, so the LFO controls themselves are irrelevant. Finally, there is no Chorus.

So how does it sound? Great, huh? Well... no. It's OK, but it sounds little like a vintage B3, being more akin to one of Hammond's transistor organs from the 1970s; the sort often observed lying unloved and unused in your Auntie Maud's living room. Nevertheless, this sound is in fact *not* far removed from that of a Hammond's unadorned tonewheel generator — it's just that it lacks the additional treatments and effects that make the Hammond A-, B- and C-series organs the sonic marvels they are. Clearly, in order to synthesize the complete sound, it's necessary to synthesize *all* the parts of the instrument.

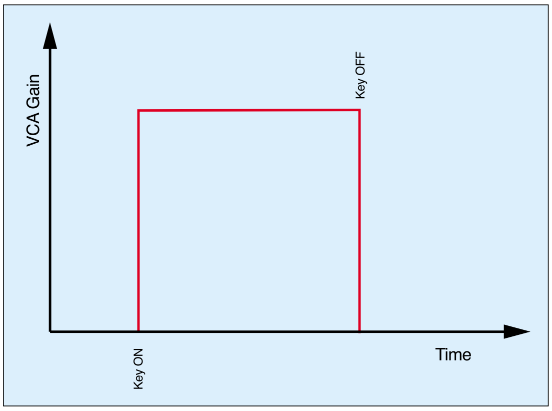


Figure 25: The resulting VCA contour.

I'll return to this point in a couple of months, but for now, I'll leave you with this thought, which may already have occurred to you — what if you don't have access to a Roland Juno 6? Can we make use of any of the principles we've learned this month on any other synth? Next month, we'll attempt to do just that. SOS

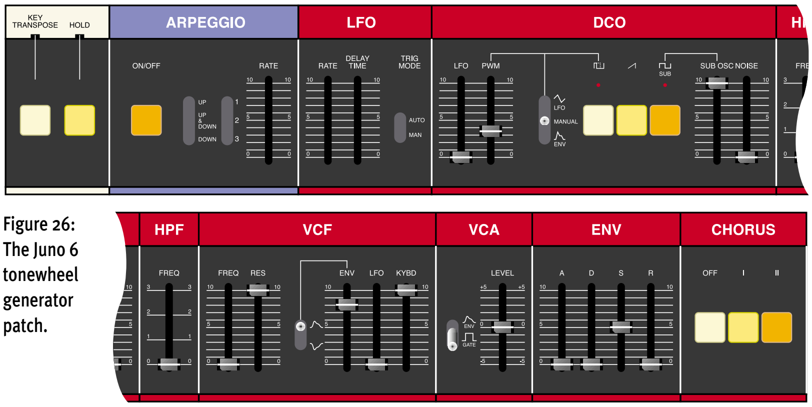


Figure 26: The Juno 6 tonewheel generator patch.

# Synth Secrets

## More On Synthesizing Tonewheel Organs

Gordon Reid

If you followed last month's advice, you'll know how to synthesize a basic Hammond tone on a Roland Juno 6. But can the same technique be applied to any other synth?

You may recall that last month, I described how, many years ago, I embarked upon a quest to find an affordable and manageable synthesizer to replace my ageing Crumar Organiser and Korg BX3. I left you with the solution I found, a Hammond patch created on a Roland Juno 6 (shown here as Figure 1 below). If you refer back to last month's article, you'll remember that the harmonic spectrum of this patch is as shown in Figure 2, where the three red squares represent the amplitudes of the first — and only — three harmonics present in the sound; the basis for a fine emulation of the 88 8000 000 registration. (For an explanation of the curious axes on the graph in Figure 2, and why they are particularly well suited to depicting the harmonic spectrum of the output from a tonewheel organ, I again refer you back to last month's instalment of this series).

Now, you might think that these diagrams reveal the secret for synthesizing all manner of Hammond registrations on the Juno. After all, the self-oscillating VCF — which is responsible for the presence of the third harmonic — can be tuned to any frequency, so in theory, we should be able to create any patch that uses three drawbars, provided that two of them, those generated by the sub-oscillator and the main DCO, are an octave apart.

For example, if we slide the VCF cutoff up to the next drawbar frequency (the fourth harmonic of the fundamental, which is the 4' drawbar) we obtain strong components at the first, second and fourth harmonics, with a reduced contribution from the third harmonic; maybe something along the lines of an 82 8800 000 registration. But when we try this, something doesn't sound quite right. While the basic tonality is much as you might expect, the patch is too bright, and lacks the character of half a hundredweight of pickups, valve preamps, and rotating steel.

Original Photo: Richard Ecclestone

To explain this, I've calculated the amplitudes of the frequencies present in the new patch, and shown the results in

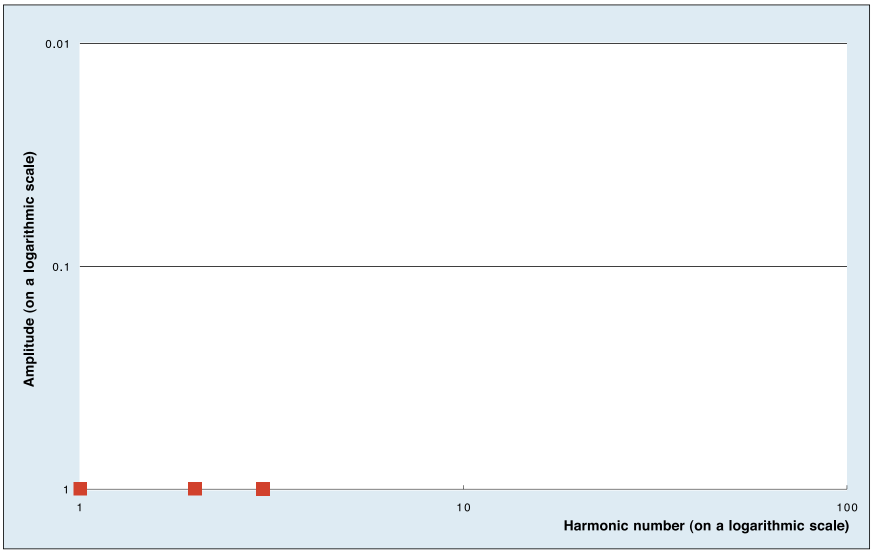


Figure 2: The harmonic spectrum of the patch shown in Figure 1.

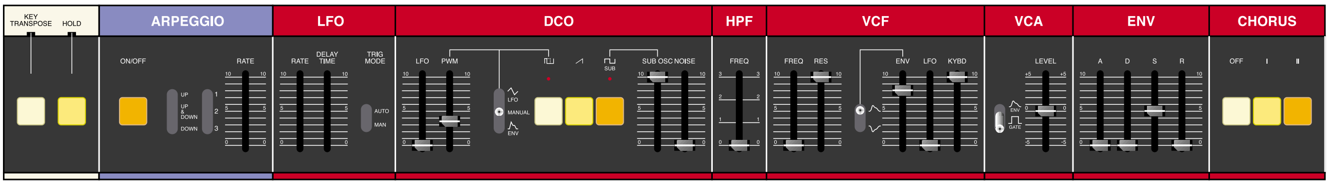


Figure 1: Last month's Juno 6 Hammond 88 8000 000 patch.

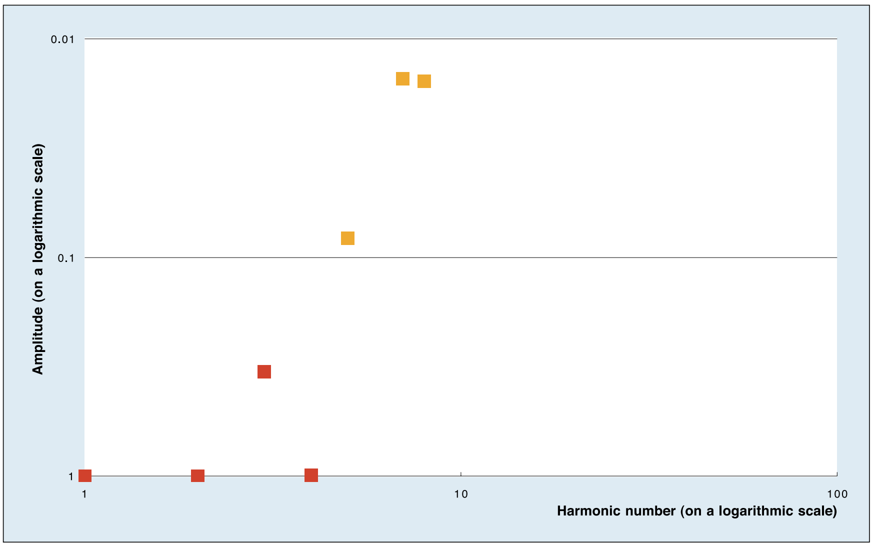


Figure 3: Trying to synthesize 80 8800 000.

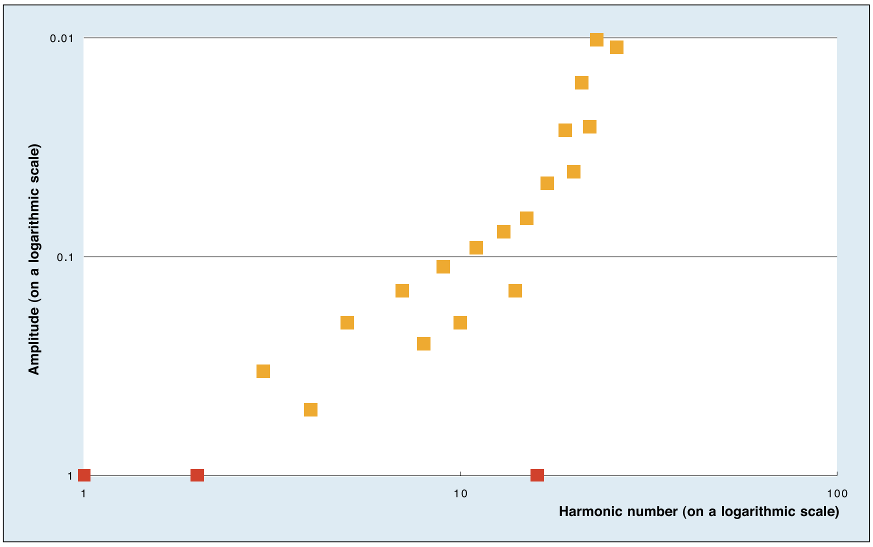


Figure 4: Failing to synthesize 80 8000 008.

▶ Figure 3 (above). As you can see, the first four harmonics are present in the expected amounts, but three more — which I have shown in orange — are also present, albeit at lower amplitudes. We might be able to excuse the rightmost of these because it lies on the eighth harmonic, and therefore represents a bit of leakage from the 2' drawbar. But the contributions from the fifth and seventh harmonics are not so welcome. They don't sound 'wrong' exactly — after all, they lie in their correct positions in a perfect harmonic series — but their contributions are inappropriate, and it is these that make the patch sound too bright (the 6th harmonic is entirely absent, but I'll leave you to work out why).

The situation deteriorates further if we raise the cutoff frequency any more. There are two reasons for this. Firstly, the filter passes all the harmonics that lie below the cutoff frequency. This is no good because, as shown last month, the idealised spectrum generated by the Hammond tonewheel engine — which goes as high as the 16th harmonic of the 16' drawbar — does not include the fifth, seventh, ninth, 11th, 13th, 14th or 15th harmonics. Secondly, the densely packed higher harmonics are not attenuated as rapidly as the widely spaced lower harmonics, so we hear many overtones above the cutoff frequency of the self-oscillating filter.

To demonstrate this dramatically, I have calculated 80 8000 008 as recreated by the 'Juno method' as described last month. This is a perfectly acceptable Hammond registration which, when patched on the synth, is a sonic mess. Again, the desired harmonics are shown in red (see Figure 4, above), and the unwanted ones in orange. As you can imagine, it sounds nothing like the real thing.

If this weren't bad enough, the tracking of the Juno's filter becomes very unstable at higher frequencies. It is superb at low multiples of the fundamental because it 'locks onto' the strong second, third and (just about) fourth harmonics, producing a pure, stable tone at pitches that relate precisely to the 8', 5 2/3' and 4' drawbars. But as the harmonic number rises, its ability to lock on diminishes, and it starts to float around. The result is a strange, tuned noise that is interesting, but nothing whatsoever to do with the sound of a Hammond organ. When it comes to the crunch, the 'Juno method' is capable only of synthesizing the 88 8000 000 registration with any degree of realism. So perhaps we should now look elsewhere to synthesize a more flexible imitation of the Hammond.

## Another Method Of Hammond Synthesis

In 1981 and 1982, Genesis were on tour promoting their *Abacab* album. For keyboard player Tony Banks, this must have been a very different experience compared with the tours of the 1970s. Gone was his Mellotron, gone was his RMI Electrapiano, and gone was his ARP Pro Soloist. And, most relevant to this month's discussion, gone was his Hammond T-series organ, to be replaced by a dual-manual Sequential Circuits Prophet 10.

As well as being hideously expensive, the Prophet was, and is, a large and heavy synthesizer, which means that it is just as much a pain in the posterior as an organ. Given that it is not particularly reliable, it seems odd that Tony should have adopted it in this fashion. Indeed, he told me many years ago that he carted two of them around on tour (with one as a spare), although he preferred not to use the second instrument because it sounded different from his favoured one. I'm not surprised... the tuning of the 20 oscillators and the 10 low-pass filters on the Prophet 10 is not what you would call 'precise'.

Nonetheless, Tony produced a fine organ sound on that tour, and the method he used illustrates a useful principle, so I thought that it would be interesting to recreate his patch.

Figure 5, which is shown on the next couple of pages, and will cause the graphics department at *Sound On Sound* to stick large pins into little Gordon effigies, shows the voice structure of a Prophet 10. It's huge, even though I've omitted the patch selection and housekeeping section of each of its two control panels for the sake of practical representation. Hang on a second... *two* panels?

Of course, the Prophet 10 has only one *physical* panel, but it really is two synths, each similar to a Prophet 5. Each has a dedicated keyboard, and each offers dual ▶

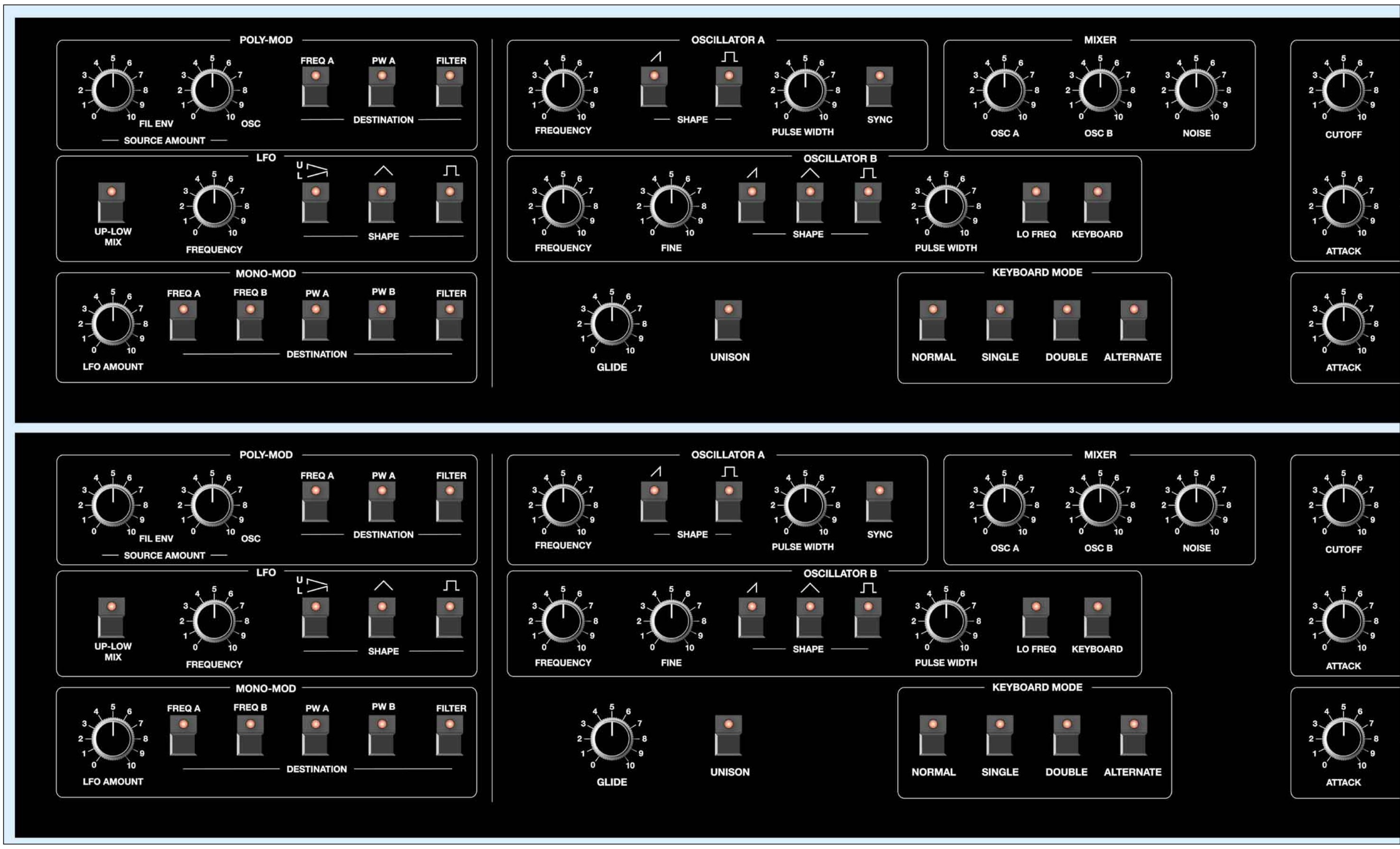


Figure 5: The voicing of the Prophet 10.

oscillators per voice, a 24dB-per-octave low-pass filter, dual ADSR contour generators per voice, an LFO, plus the Prophet 5's renowned Poly-Mod section.

In Normal mode, the Upper synth is played from the upper manual, while the Lower synth is played from, well... the lower one. This means that we can take either one, and patch it using the 'Juno method'.

We'll start with the VCOs. Figure 6 (below) shows that I have selected a pulse wave for Oscillator B, and set the pulse width to '5', which is the setting at which it produces a square wave. You'll also see that I have tuned it to the lowest pitch available, with no fine tuning offset, and that the 'Keyboard' LED is lit, which shows that the oscillator will track the keyboard in a conventional manner. Oscillator B is, therefore, performing the same task as the sub-oscillator in the Juno patch.

Oscillator A is also programmed to produce a pulse wave, but on this occasion, the pulse width is 33.33 percent, just as it was last month. The Frequency knob is tuned by ear to produce a pitch that is precisely one octave above Oscillator B. Once this is set correctly, we can adjust the relative amplitudes of the oscillators (which are, in effect, the drawbar settings of the 16' and 8' pitches) in the Mixer.

Next comes the filter section (shown in Figure 7, left). Firstly, the 'Keyboard' switch must be on (ie. with the red LED lit) so that the filter tracks the keyboard. Then, as with the Juno patch, we set the cutoff frequency so that it lies precisely on the 5 2/3' pitch, and increase the resonance until the filter begins to oscillate and produces a sine wave.

The Prophet 10 has a dedicated ADSR contour generator for the filter, and I have set all its knobs to zero. This is because the P10's envelopes are not the snappiest in the world, and we need to use the minimum settings to obtain the 'key-click' sound (see Figure 8, above) at the start of each note, as explained last month. You determine the amount of click by adjusting the Envelope

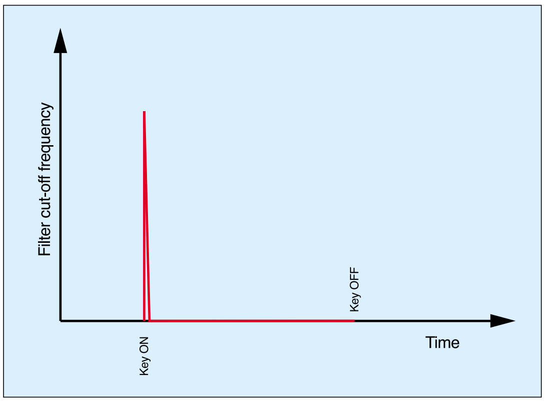


Figure 8: Creating the 'key-click' sound using the filter cutoff frequency contour.

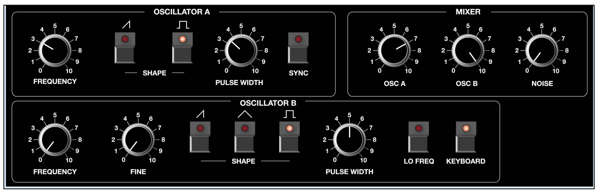


Figure 6: Setting up the Prophet 10's dual oscillators.

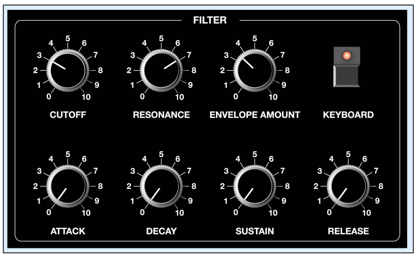
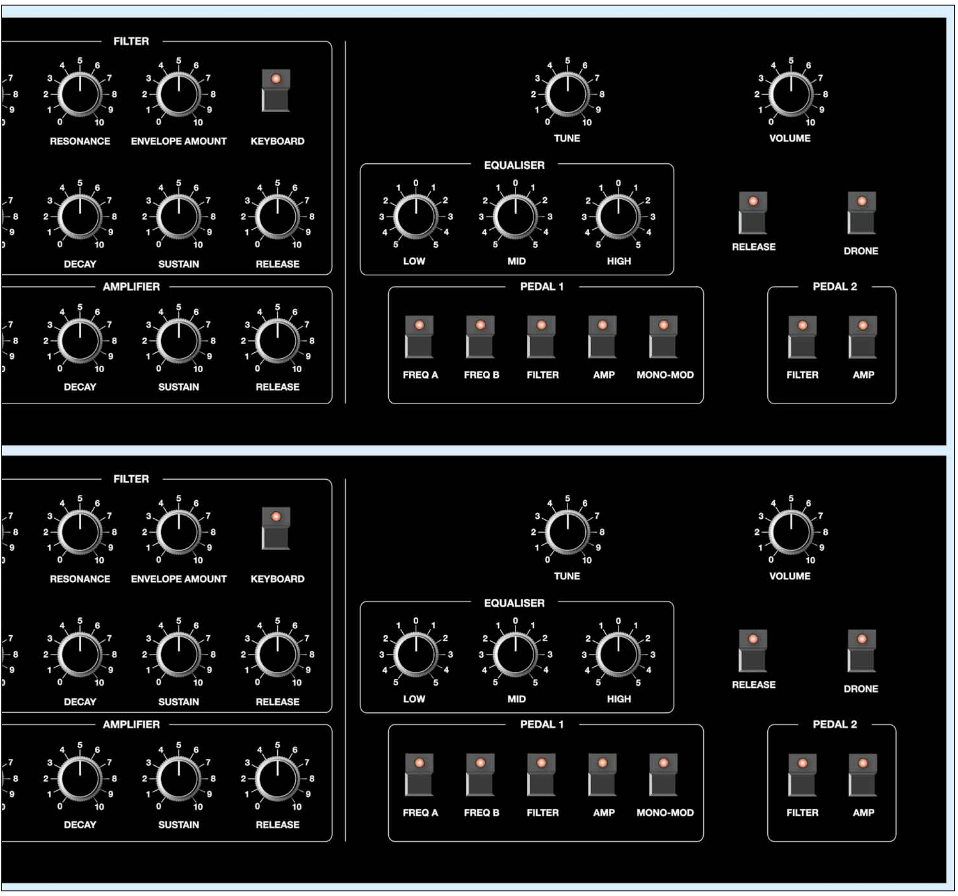


Figure 7: Setting up the filter.



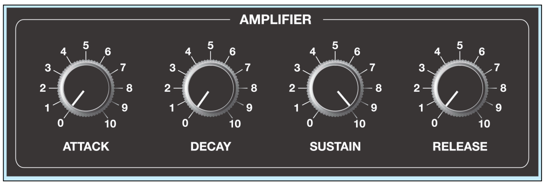


Figure 9: The 'organ' amplitude envelope settings.

Amount knob to taste.

The next part of the patch is easy. We want an 'organ' envelope, so we can set the amplitude ADSR as shown in Figure 9, with instantaneous Attack, maximum Sustain level, and no Release. The Decay segment of this contour is, of course, irrelevant (see Figure 10, below).

And there you have it: defeat all the modulation sources and you have programmed the wonderful Prophet 10 organ patch, a gorgeous sound from one of the greatest synthesizers ever built. Fantastic! Or is it? For one thing, the Prophet 10 hardly answers to my required description, 'affordable'. And then there's the sound itself. Sure, it's nice, and has a warmth and presence that you would be proud to use, especially if you use the onboard EQ section to boost the middle frequencies. But, just as I suggested last month, for this purpose the Prophet is still the inferior of the vastly cheaper Juno. Why?

The answer lies in the aforementioned instabilities of the Prophet 10. Despite the microprocessor that lies at its heart, it is a truly analogue synth, and you

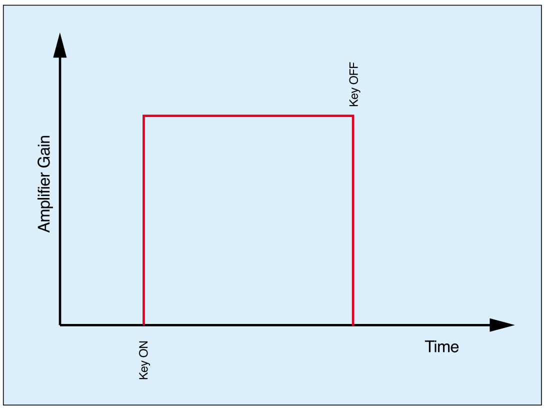


Figure 10: The amplitude contour.

can press the Tune button until you get blisters, but you still won't get its oscillators in perfect tune, much less its filters. What's more, the quantisation of the controls is very apparent, and this makes it impossible to use the 'Juno method' effectively. Back in Part 21 of this series (see *SOS* January 2001, or surf to www.soundonsound.com/sos/jan01/articles/synthsec.asp), we discussed the reasons why analogue synths with memories must have quantised controls. So, while you might be able to tune any one of the Prophet's filters to precisely the correct pitch, the one in the next voice might be far enough removed that, when you turn the filter knob a tiny amount to bring it into line, the cutoff frequency jumps so far that the situation is worse than before. On my Prophet 10, one of the filters on the Upper keyboard is always a few cents out of tune, and, while it tracks correctly, the voice that contains it sounds significantly different from the other four. You may choose to call this 'analogue warmth', but it's not. It's just plain wrong.

## Two Is Better Than One

So how can we overcome this? The old Genesis videos demonstrate that Tony's Prophet 10 was capable of a much better likeness to the old Hammond, so there must be a way... And there is.

The secret lies in the 'two synths in a box' nature of the big Prophet, and the four keyboard modes that it offers. Up until now, I've been assuming that we've been in Normal mode which, if the 'Juno method' had been successful, would have allowed me to create different patches for each keyboard, and to play the Prophet 10 as a dual-manual organ, or as a single-manual organ plus a string ensemble, or brass section, or whatever. But the method was *not* successful, so now I'm going to place the synth in Double mode (see Figure 11, below). This allocates Upper Voice 1 and Lower Voice 1 to the first note you play, Upper Voice 2 and Lower Voice 2 to the second note you play... and so on. In other words, I have placed both synths under the control of one keyboard (the Prophet 10 was one of the first instruments to offer layering). This makes it possible for us to patch registrations containing four pitches, and without having to use the filter as an oscillator.

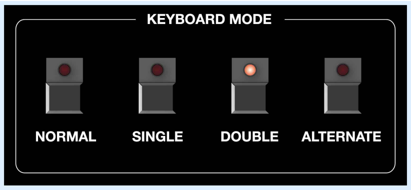


Figure 11: The Prophet 10's four voice-allocation modes.

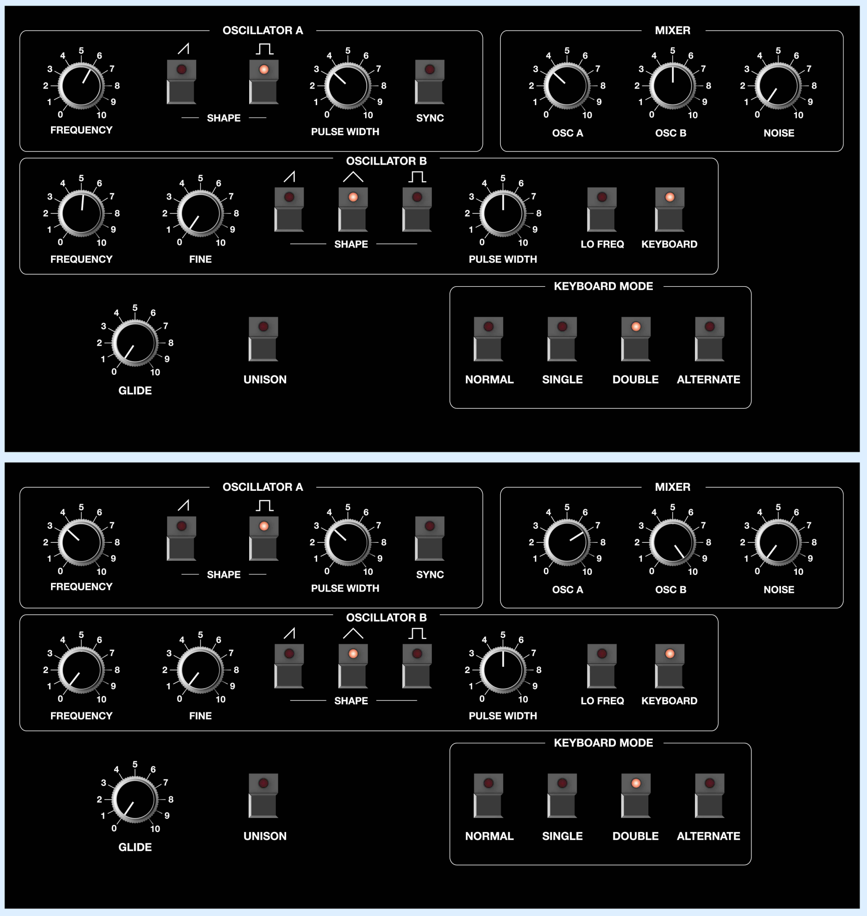


Figure 12: Using four oscillators to emulate four drawbars.

Consider Figure 12 (above). This shows the Upper and Lower Oscillator and Mixer sections simultaneously, with Double mode selected, and all four oscillators tuned and balanced to produce the 88 8800 000 registration. The Lower section is identical to Figure 6, with one exception: I have switched off the square waveform in Oscillator B (the oscillator producing the 16' pitch) and selected the triangle wave instead. This is as close as the Prophet will come to emulating the (near) sine wave produced by a tonewheel generator. The Upper section is set up using the same waveforms, but tuned so that oscillators B and A produce the pitches of the 5 2/3' and 4' drawbars respectively.

Of course, there's nothing forcing us to use these pitches, and we no longer have to use the self-oscillating filter to produce the 5 2/3' pitch. So, in this way, the Prophet 10 patch is superior to the Juno's.

Now we must reprogram the filter sections for both synths, eliminating the resonance, but keeping the cutoff low enough to attenuate the unwanted harmonics generated by the triangle and pulse waveforms (see Figure 13, right). The ▶

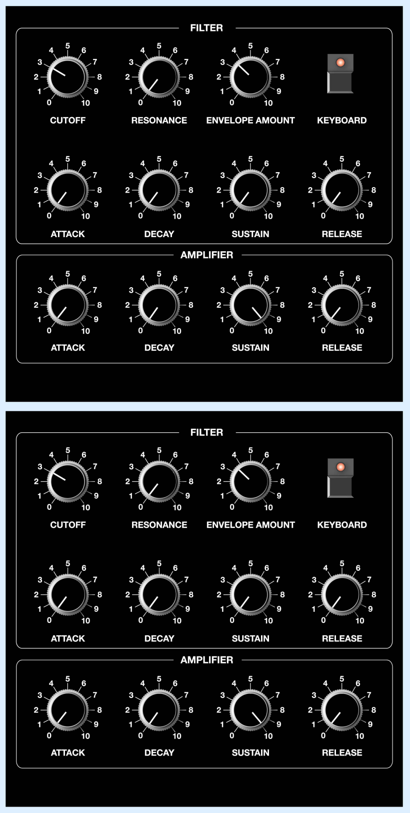


Figure 13: The filter and amplifier settings for a four-oscillator Hammond emulation.

amplifier ADSRs should, of course, be identical to each other and to that shown in Figure 9, and all modulation should be defeated. Having set all of this, we should now be able to create and play any registration, provided that it uses only four drawbars at a time.

Nevertheless, the Juno still sounds the better of the two. Far from being the millstone that some anoraks would have you believe, the precision offered by its digitally controlled oscillators and its superb filter tracking ensures consistency across all the notes played, and this is exactly what a Hammond patch requires.

## A Better Mousetrap?

I don't know about you, but I feel decidedly uneasy that the Juno has outshone the mighty Prophet. Nevertheless, this set me thinking... there must be a synth that's not too expensive, but which combines the stability and tuning accuracy of the Juno's DCOs and filter, and also offers the flexibility of four oscillators and dual signal paths.

Of course there is! It's the Roland JX10, which has a the 'two synths in a box' architecture, but is digitally controlled. Surely this is the best of both worlds, and must sound superb? Well... no, it doesn't. I used a JX10 as my main stage keyboard/controller for more than a decade, and after numerous abortive attempts, I never again attempted to use it for organ patches. Experience showed that JX10 organ patches are at best unconvincing, and that's perhaps the reason that my Juno 60 survived as a gigging instrument for as long as it did.

Hmm... what other affordable analogue synths can we try? The Oberheim OB-series? Far too inaccurate. A Memorymoog? You've *got* to be joking... How about the Prophet 600, or the Korg PS3200, or the Crumar Bit One, or the Akai AX60, or the... Stop it Gordon, take a deep breath, and relax. None of these fit the bill. When it comes down to it, the Junos really are remarkable little synths, and it is no wonder that they often sound superior to instruments worth many times as much.

Nonetheless, there is at least one low-cost analogue/digital hybrid does an even better Hammond emulation. The sound quality is superb, and it is completely flexible, being capable of any of the 387,420,489 registrations that you care to name (did anybody check my maths?). Yet its second-hand value is close to zero, and you would probably walk past one if you saw it in a car-boot sale. It's one of my favourite synths of all time. It's the Kawai K3 (shown above right).

## It All Adds Up

Last month, I mentioned that dedicated additive instruments such as the Kurzweil 150, Kawai K5 and Kawai K5000 were capable of some fine Hammond sounds, as are the larger DX-series FM synths. But the thing that makes the K3 special is its combination of a primitive form of additive synthesis plus one of the scrummiest analogue filters ever designed by man, the SSM2044.

| HARMONIC NUMBER | 1 | 2 | 3 | 4 | 5 | 6 | ... up to 128 |
|---|---|---|---|---|---|---|---|
| VALUE | 31 | 31 | 31 | 0 | 0 | 0 | ... all zero |

Unlike the dedicated additive instruments mentioned above, the additive section in the K3 allows you to create just one spectrum (and, therefore, waveform) at a time, but this comprises up to 32 partials distributed anywhere among the first 128 harmonics of the pitch. Setting this up is a doddle; you just select the harmonic number and dial in an amplitude between zero and 31. Simple!

This means that we can construct any conventional registration using the first, second, third, fourth, sixth, eighth, 10th, 12th and 16th harmonics, or reproduce the extended drawbar set offered by a handful

| KAWAI K3: HAMMOND 88 8000 000 REGISTRATION | | | |
|---|---|---|---|
| OSC 1 | | | |
| • 1 | Wave | 32 | The additive wave |
| • 2 | Range | 16' | |
| • 3 | Portamento Speed | 0 | No portamento |
| • 4 | Balance | -15 | Only OSC1 used |
| • 5 | Pitch Bend | 0 | |
| • 6 | Auto Bend | 0 | |
| OSC2 | | | |
| • 7 | Wave | n/a | |
| • 8 | Coarse Freq | n/a | |
| • 9 | Fine Freq | n/a | |
| FILTER | | | |
| • 10 | Cutoff | 65 | Sounds about right |
| • 11 | Resonance | 0 | |
| • 12 | Low Cut (HPF) | 0 | No high-pass filtering |
| • 13 | Env Amount | 31 | Maximum amount |
| • 14 | Attack | 0 | |
| • 15 | Decay | 0 | A fast key-click 'blip' |
| • 16 | Not used | | |
| • 17 | Sustain | 0 | |
| • 18 | Release | 0 | |
| AMPLIFIER | | | |
| • 19 | Level | 31 | Maximum amplitude |
| • 20 | Attack | 0 | |
| • 21 | Decay | 0 | |
| • 22 | Not used | | |
| • 23 | Sustain | 31 | A 'square' amplitude contour |
| • 24 | Release | 0 | |
| LFO | | | |
| • 25 | Shape | n/a | |
| • 26 | Speed | n/a | |
| • 27 | Delay | n/a | |
| • 28 | Oscillator Amount | 0 | |
| • 29 | VCF Amount | 0 | |
| • 30 | VCA Amount | 0 | |
| TOUCH SENSITIVITY | | | |
| • 31 | Velocity -> VCF | 0 | |
| • 32 | Velocity -> VCA | 0 | |
| • 33 | Pressure -> OSC Balance | 0 | |
| • 34 | Pressure -> VCF | 0 | |
| • 35 | Pressure -> VCA | 0 | |
| • 36 | Pressure -> LFO OSC Amount | 0 | |
| KEYBOARD TRACKING | | | |
| • 37 | VCF | 9 | Approximately 100-percent tracking |
| • 38 | VCA | 0 | |
| CHORUS | | | |
| • 39 | Chorus | 0 | Off |

of rare Hammonds, or even imitate the 'EX' mode of the new, DSP-driven Korg CX3 and BX3 emulators. Think about it; we no longer need to resort to trickery to obtain the spectrum in Figure 2. We simply select harmonic #1 and give it an amplitude of 31, select harmonic #2 and give it an amplitude of 31, select harmonic #3 and give it an amplitude of 31, and then press the 'Write' button to calculate the waveform, as shown in the smaller value table opposite. We then reduce the filter cutoff frequency a little to remove some stray upper frequencies that, in a perfect additive world, wouldn't be there in the first place, and the result is...? Superb.

We construct the rest of the patch exactly as before, with a 'spitty' filter contour as shown in Figure 8. We do this by setting the ADSR values for the filter (parameters 14, 15, 17 and 18) to be 0, 0, 0, 0... which is the same as the Prophet 10 knobs shown in Figure 7, but represented in numerical form in the K3's 'digital parameter access' user interface. Likewise, the amplitude ADSR (parameters 20, 21, 23 and 24) is set to 0, 0, 31, 0... the same as the knobs in Figure 9, and therefore defining the 'square' amplitude envelope of Figure 10.

Next, we defeat the velocity sensitivity and pressure sensitivity (neither of which are appropriate for a Hammond patch), reduce all the modulation amounts to zero, and... bingo! The complete patch is shown in the large table opposite.

So there we have it... We started with the little Juno, which is cheap and cheerful, and synthesizes just one Hammond registration extremely well. We then graduated onto the mighty Prophet 10, which is far from cheap, but is limited to four 'drawbars' and — unless every voice is tuned absolutely precisely — produces no meaningful Hammond registrations well. Finally, we ended up programming an almost unknown, valueless analogue/digital hybrid. Yet it is this that is best suited to Hammond emulation, which proves to be the most flexible, and which has produced the most satisfying result. You might think that I've cheated by introducing additive synthesis (and you would probably be right) but given that my original aim was to program convincing registrations on something that was cheap and physically light, but sounded as good as a Korg BX3, I'm happy. The answer, ladies and gentlemen, is the Kawai K3.

## Epilogue

As with last month's Juno patch, and despite what I've just written, the K3 patch described here doesn't sound all that much like a real Hammond. As I explained last month, this is because these patches make a good fist of synthesizing the sound of an *unadorned* tonewheel generator — as yet, I've made no attempt to reproduce the chorus/vibrato, percussion, and overdrive effects that really 'make' the Hammond sound. Next month, we'll do what we can to emulate these, and see whether we can use the Juno to produce entirely convincing imitations of the big Hammonds. Until then... enjoy your organ! SOS

# Synth Secrets

## Synthesizing The Rest Of The Hammond Organ — Part One

Gordon Reid

**So, you can synthesize a Hammond's tonewheel generator — but what about its all-important effects? This month, we look at recreating the Hammond's percussion, vibrato, overdrive, and reverb — and find that it's harder than you might think...**

I find that my relationships with my synths can be much like any other romantic entanglements... fun and frustration in turns. When you're lucky, everything comes naturally, and you attain what you crave both easily and quickly. On other occasions, you have to work hard at things, and sometimes you just have to give up, pretending that you weren't that interested in the first place.

For the past two months, I think that it's fair to say that this series has been dishing up a good deal of the former, with the basis for some fine tonewheel organ patches being produced on some unlikely synths. But, as I wrote when I left you last time, what these have all lacked is the excitement introduced by the Hammond's effects and side-effects; percussion, chorus/vibrato, leakage, and overdrive. So now, we're going to attempt to spice things up still further. Unfortunately, as in real life, some relationships start out as fun, but lead to frustration, although you usually learn some important lessons on the way. In this case, even though we don't necessarily achieve everything we set out to do, there's plenty to be learned about how a tonewheel organ creates its distinctive sound along the way.

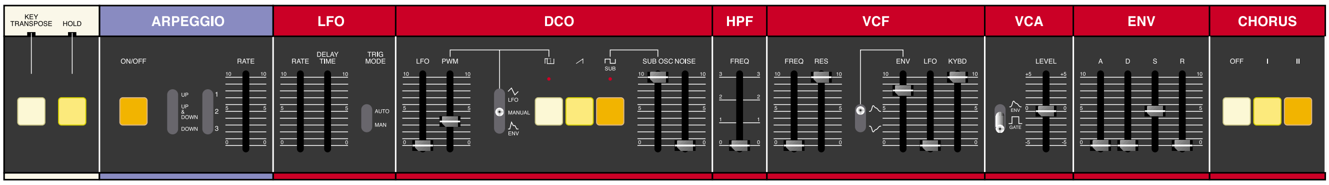


Figure 1: Returning to the Juno 6 Hammond patch.

### Matching Registrations

Just across the room from where I'm writing, there sits one of greatest organs ever crafted by human hands: a Hammond A100, an instrument every bit the equal of the B3 and C3. If you're unacquainted with Hammond genealogy, let me explain...

For many decades, the company had a policy that its 'spinet' organs (those with four-octave keyboards) had built-in speaker systems, while the larger 'console' organs (those with five-octave keyboards) required external speakers, or 'tone cabinets'. Sometime after the launch of the B3 and C3 in 1955, Hammond's customers made it clear that they wanted a self-contained organ with the wonderful sound of the new flagships, but also the reverb and internal speakers of the less expensive spinets. Thus was the A100 born:  a B3/C3 tonewheel generator and controls mounted inside a smaller case that nonetheless includes a spring reverb, dual valve amplifiers and three chunky speakers.

So close is the relationship between the

Original photo: Richard Ecclestone

B3, C3 and A100 that there is nothing to stop you from sliding the tonewheel generator out of one and wiring it into the others (well, nothing other than a few hundred wires!). This means that the A100 is the superior of the three organs, because it sacrifices nothing, but takes up less room and adds the reverb and speaker system. This superiority is not borne out by the second-hand prices of these models, which baffles me, but there it is.

Anyway, having the Hammond sitting just a few feet from my Juno 60 makes it simple to investigate and resynthesize each of the Hammond's effects. So, to start, I'm going to match the sounds of the two instruments such that applying the same effect to each should yield the same result.

I do this by switching off the percussion and chorus/vibrato effects on the A100, limiting the volume somewhat, and making sure that I don't play the result through the attached Leslie rotary-speaker system. Now, if I play the Juno patch that I developed two months ago (see Figure 1) through a high-quality amplifier/speaker combo, while simultaneously playing the Hammond through its own speakers, the similarity is almost uncanny, *provided* that I match the Hammond's registration to imitate the Juno. This is necessary because — despite my best efforts two months ago (see www.soundonsound.com/sos/nov03/articles/synthsecrets.htm) — the synthesizer

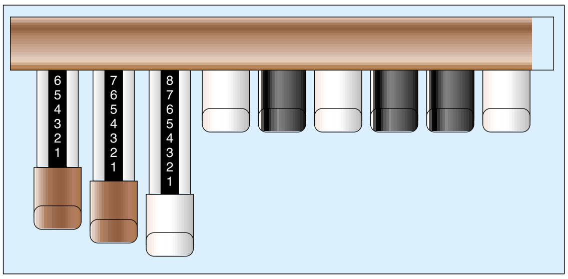


Figure 2: The Juno patch lacks the depth of 88 8000 000, lying closer to 67 8000 000.

patch is not quite a true emulation of 88 8000 000.

Most obviously, the amplitudes of the three primary harmonics lie closer to a Hammond registration of 67 8000 000, with the 8' pitch most audible, and lesser contributions from the 5 2/3' and 16' pitches, as shown in Figure 2 (above).

This result suggests that, by using the filter to synthesize the 5 2/3' drawbar, we are sacrificing some of the amplitude of the sub-oscillator. This is not altogether surprising. In fact, it is exactly what we would expect from most analogue filters, because high filter resonance usually suppresses lower frequencies, as shown in Figure 3, above.

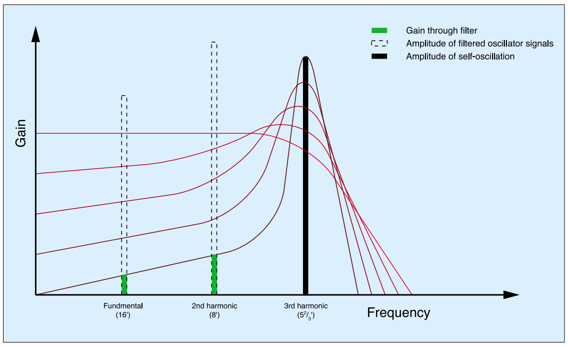


Figure 3: Increasing the resonance of most filters reduces the low-frequency amplitudes of low-frequency signals passing through them.

Listening more closely reveals that the Juno not only lacks the low-frequency 'oomph' of the Hammond's 88 8000 000 registration, but is also a tad brighter. As a result, a touch of the next two or three Hammond drawbars makes the two instruments sound even more similar. After a few minutes' comparison, I found the registration 67 8321 000 to be about right (see Figure 4, below).

Again, this is not surprising. After all, we would not expect the Juno filter to eliminate everything above the cutoff frequency, even when oscillating. This explains the need for the low-level signals injected by the 4', 2 2/3' and 2' drawbars.

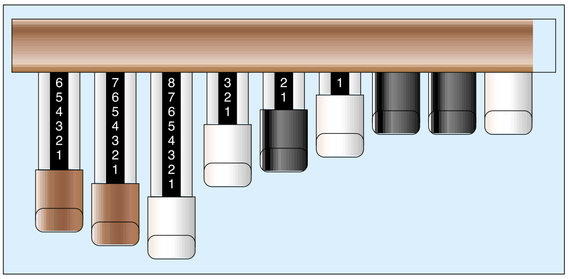


Figure 4: The registration 67 8321 000 is much like the Juno patch.

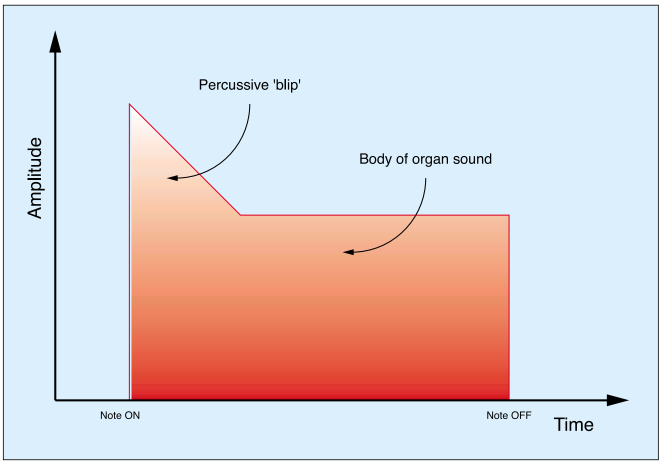
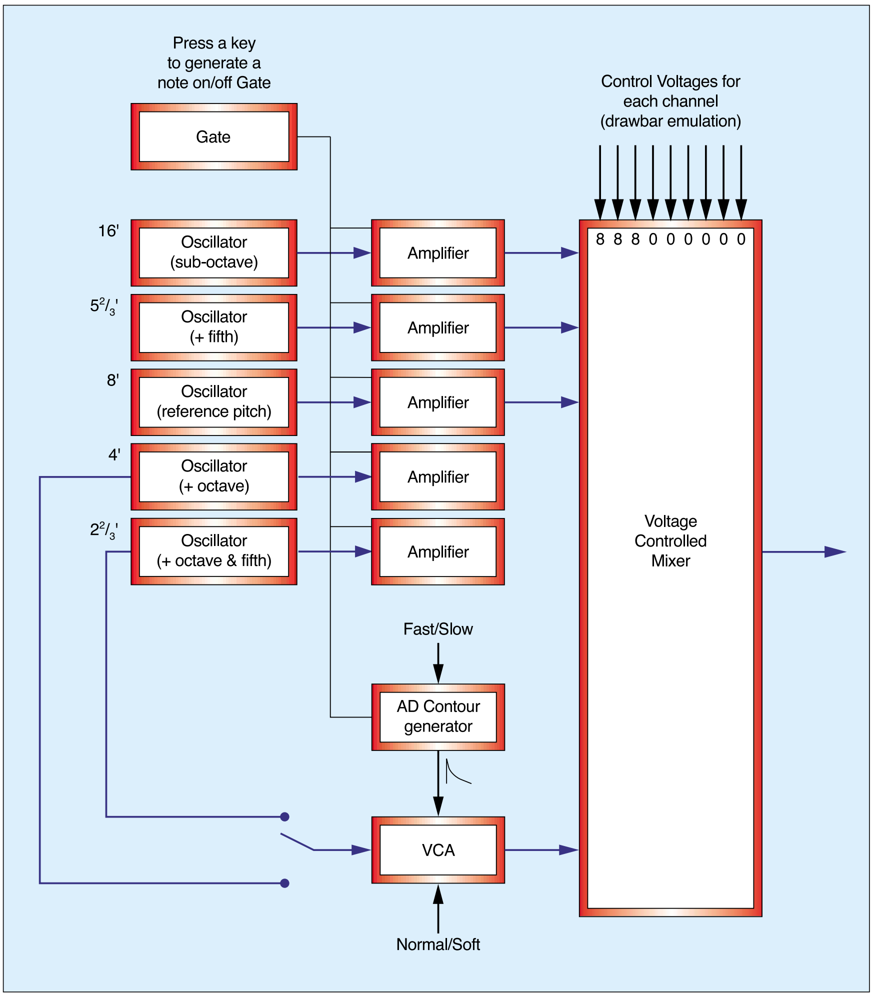


Figure 5 (left): The percussive 'blip'.

Figure 6 (below): Creating second-harmonic and third-harmonic percussion using modules.



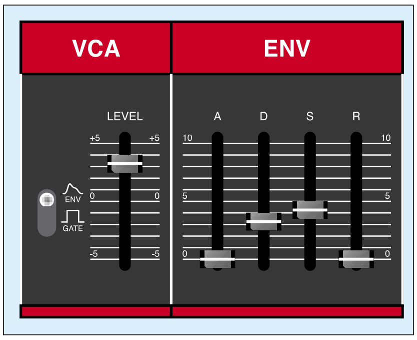


Figure 7: Adding a percussive shape to the amplitude contour.

Anyway, having matched the sounds of the two instruments, we're now in a position to move on to...

## Hammond Effects — Percussion

A Hammond's percussion has nothing to do built-in rhythm units. That is, there *are* Hammonds with such units built in, but when I say Hammond percussion, I'm not talking about them. No, the four percussion controls on an A100 allow you to add a greater or lesser amount of either the second or third harmonic of the 8' pitch — ie. of the 4' or 2 2/3' drawbar — as an accent at the start of the note. The amplitude 'shape' of the result is therefore as shown in Figure 5. It's worth pointing out that adding percussion also reduces the loudness of the sustained part of the note, but we're going to overlook this. Likewise, Hammond percussion is polyphonic, but of the single-triggering variety, so if a previous note is held, the percussion does not sound. Again, we'll overlook this, because trying to recreate it would take us into areas best not trodden in an article of this length.

Returning to the four percussion controls on the A100, the On/Off switch is self-explanatory, as is the Second/Third selector. This leaves just the Normal/Soft and Fast/Slow switches that control the loudness and decay rate of the effect. Simple though these seem, to emulate all their combinations would stress the resources of any analogue synth. Nonetheless, if we had the resources of a suitably expansive synth to hand, we could set up a patch to produce just one organ note, imitating the percussion by diverting part of the 4' or 2 2/3' signal through a VCA controlled by an AD contour generator. I have shown a stylised representation of this (using 88 8000 000 as the basic registration and omitting unused footages) in Figure 6. Complex, isn't it?

Unfortunately, the Juno does not offer the complexity needed to imitate the structure in the diagram. Faced with these limitations, many synth programmers attempt to give the *impression* of percussion by modulating the audio VCA to create the amplitude blip shown in Figure 5. On the Juno, you would obtain this by flipping the VCA switch from 'Gate' to 'Env', and by adding a little Decay to the ADSR contour. I have shown these changes in Figure 7 (below).

This creates the audio effect shown in Figure 8, which is far removed from the true percussion effect represented by Figure 9 (both at the top of the next page). What's more, the patch in Figure 1 creates key-click by using the ADSR to modulate the VCF cutoff frequency. The extended decay in Figure 7 changes this click into a completely un-Hammond-like soggy squelch. So, if we want to use this idea, we must disconnect the filter from the envelope generator and retune the cutoff frequency so that it again gives us the 5 2/3' drawbar pitch (see Figure 10 overleaf).

Of course, our failure to synthesize even a basic percussion effect is not indicative of a limitation of analogue synthesis in general, and things are much more promising if we move away from the Juno, and consider a more complex synth with multiple signal paths.

You may remember that the Sequential Prophet 10 introduced last month offers two

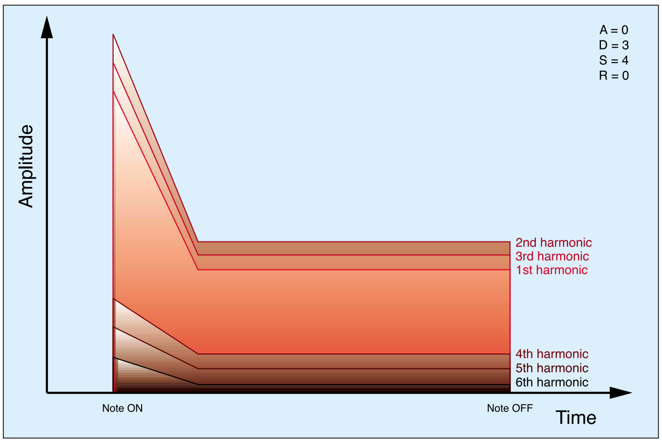


Figure 8: Using the ADSR to create a blip at the start of the note.

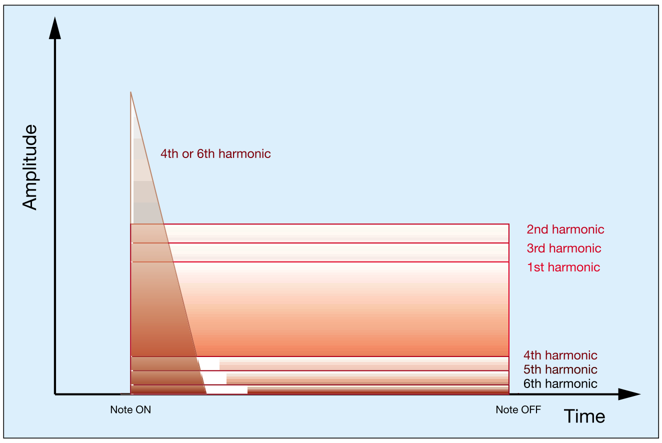


Figure 9: The Hammond percussion sound.

paths that we could use to generate any four drawbar footages of our choosing. For example, we could use the Lower synth to produce the 16' and 2 2/3' pitches, and the Upper synth to produce the 8' and 5 2/3' pitches. This allows us to use the Lower filter to create a percussive 'blip' at the front of notes, controlling the loudness of the 2 2/3' pitch without affecting the amplitude of the other pitches (see Figure 11, below).

Figures 12 and 13 (overleaf) demonstrate why this works so well; the 5 2/3' and 8' pitches are not passing through the Lower filter, and the 16' pitch is far enough removed from the cutoff frequency to be unaffected by the changes. OK, I'm cheating, because the Prophet 10 cannot produce the sine waves needed to make this picture strictly accurate, but the result nonetheless sounds surprisingly authentic. Neat, huh?

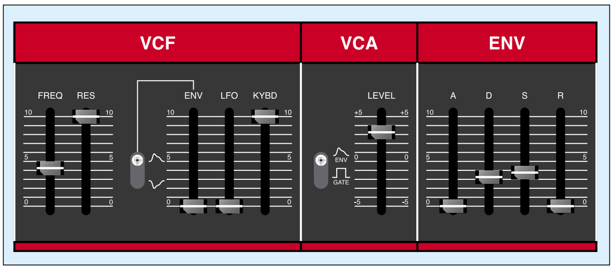


Figure 10: A useable percussion patch — but it won't fool you.

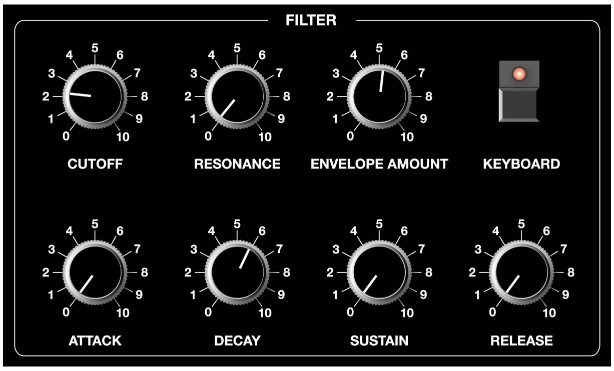


Figure 11: Using one of the Prophet 10's filters to create a far more accurate percussion sound.

## Hammond Effects — Chorus/Vibrato

Given that there's no way to emulate the Prophet's percussion settings on the Juno, let's now ignore this effect, throw a temper tantrum, and — as suggested at the start of this article — decide that we never wanted it, anyway. Instead, let's move on to the wonderful chorus/vibrato provided on the larger Hammond organs.

Chorus was not a feature of Laurens Hammond's earliest instruments, but he soon decided that the sound of his tonewheel generator was too pure, and that it needed something to impart life and movement. Some of his earliest production organs used two ranks of tonewheels detuned by a small amount to create what was possibly the world's first example of 'polyphonic oscillator detune', while some of his 'X-series' speaker cabinets had a rotor at the top of the assembly that added amplitude modulation. But Hammond wanted something with more animation, and in 1945 he designed an electromechanical device that created the pitch modulation he wanted. He called it a 'scanner' vibrato.

This uses a tapped delay line which, if we look closely at the electronics, is a type of phase-shifter constructed from low-pass filters. The signal generated by the tonewheels is applied to the input of the delay line, and a rotating pickup driven by the tonewheel generator picks the signal off the delay line at each of the tap points, one at a time. The scanner is wired so that it moves from one end of the delay line to the other, and back again, during each rotation. As it does so, the pitch shifts up and down... which is, of course, vibrato. Careful analysis shows there is also a small amount of amplitude modulation as the scanner sweeps round the taps, but we should be able to ignore this.

If you select one of the 'V' settings on the Hammond, all of the audio is routed through the scanner, and the signal suffers unadulterated pitch modulation at one of three depths called V-1, V-2 and V-3 (see Figure 14 on the next page). If you select a 'C' setting (C-1, C-2 or C-3), the output from the scanner unit is mixed with the unaffected output from the tonewheel generator, and the result is what we call 'chorus' (see Figure 15, also overleaf). This is the key to the best Hammond sounds yet, despite its apparent simplicity, only a couple of Hammond emulators manage to get it right.

So, what hope do we have of getting the Juno's onboard chorus unit to imitate the C-3 setting favoured by many organists? None, I'm afraid. The Hammond chorus mixes the straight-through signal with just a single instance of the pitch-modulated signal, so the Roland's three-stage chorus/ensemble is far too lush.

It's little consolation that we can use the Juno's LFO to create vibrato of an appropriate depth and speed... it doesn't sound the same as the Hammond's. If you want to try this, you must select the LFO rate very carefully — I find that 'six and a bit' is correct on my Juno 60 — and set the LFO depth in the DCO to create the correct amount of modulation. But this is only half the story. The 5 2/3' pitch is being generated by the VCF, so you must also raise the LFO depth in the filter section, and try to ensure that identical amounts of modulation are applied to the DCO and the VCF. If you don't, the 16' and 8' pitches will deviate more (or less) than the 5 2/3' pitch, which leads to some very unconvincing

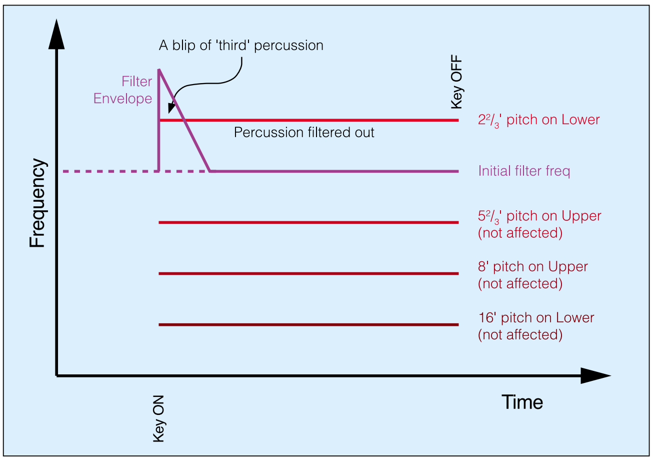

A blip of 'third' percussion
Filter Envelope
Percussion filtered out
Key OFF
$2^2/_3$' pitch on Lower
Initial filter freq
$5^2/_3$' pitch on Upper (not affected)
8' pitch on Upper (not affected)
16' pitch on Lower (not affected)
Frequency
Key ON
Time

Figure 12: Creating a percussive 'blip' using the Upper filter envelope.

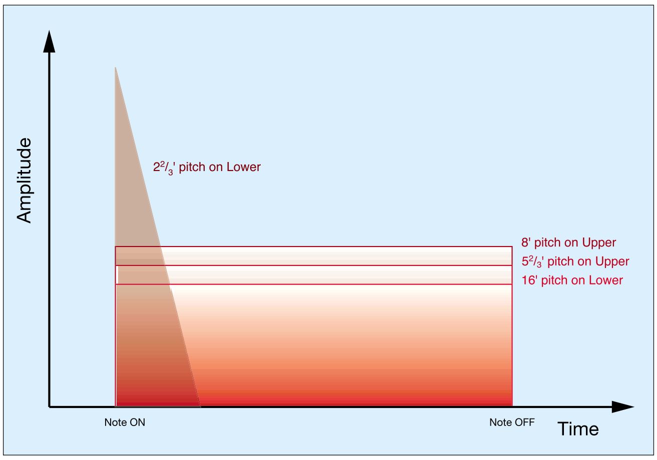


Figure 13: Hammond percussion recreated on the Prophet 10.

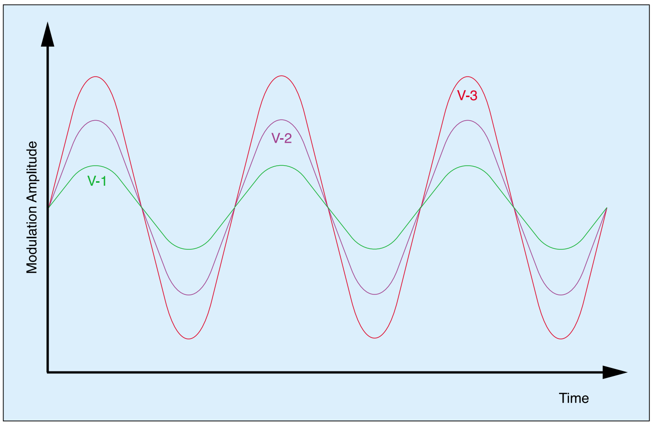


Figure 14: Three levels of simple vibrato.

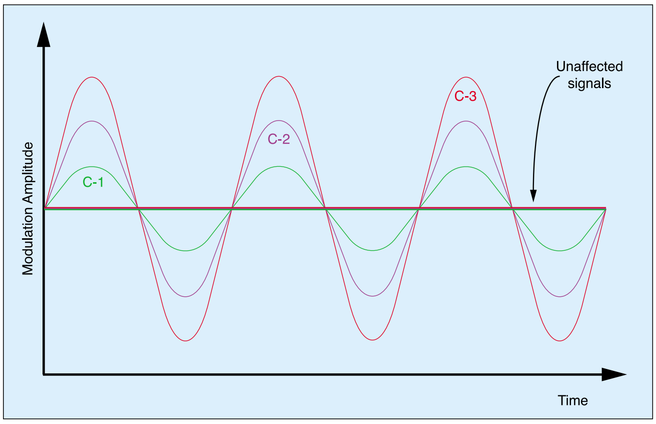


Figure 15: Three levels of chorus.

effects. I have shown the modified parts of the patch in Figure 16 (below).

To be honest, I think that these changes have turned my original Hammond patch from prime steak into dairy produce. In other words, a patch that was previously meaty now sounds cheesy. It may be theoretically accurate, but that doesn't mean that I have to like it. In fact, I never use any of my A100's 'V' settings, so I'm going to abandon the changes in Figure 16 and return, yet again, to Figure 1.

## Hammond Effects — Leakage

Another characteristic of the tonewheel generator (which, like key-click, Laurens Hammond considered to be a fault) is 'leakage', a mixture of drawbar pitches and noise that gives the A100 a characteristic, throaty quality.

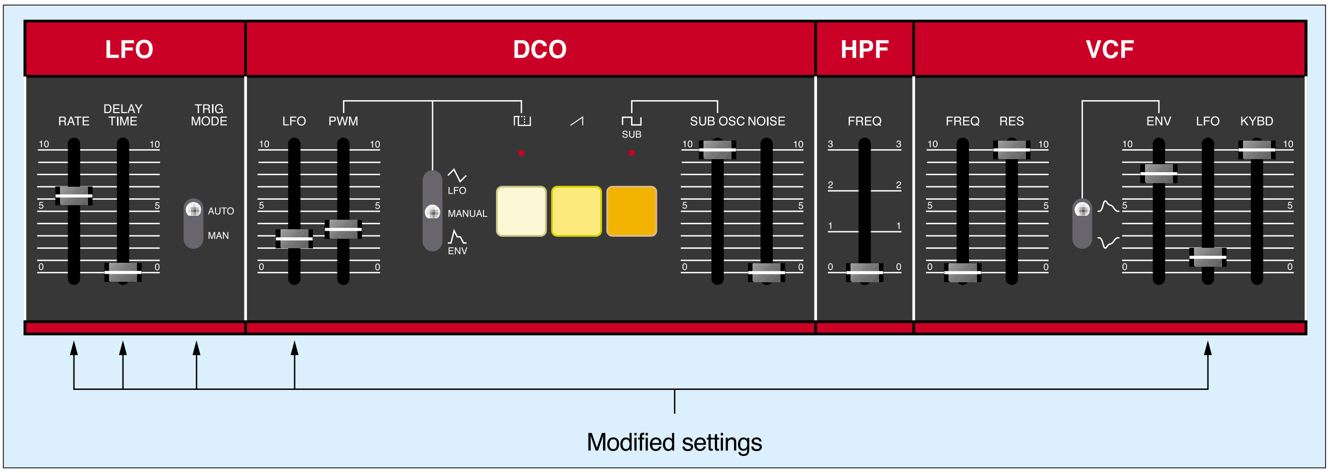
On some synths, adding the tiniest amount of noise helps to create this impression. On the Juno, however, the noise passes through the self-oscillating filter, and emerges tuned to the 5 2/3' pitch. Bah!

Because its filters are not oscillating (indeed, have zero resonance), adding noise works far better on the Prophet 10. But on consideration, I think that I'll leave well alone. Back to square one (or, to be precise, Figure 1) again!

## Hammond Effects — Overdrive & Compression

The next thing we need to consider is overdrive; our ability to cause the valve preamplifier and amplifier(s) in the Hammond to distort. Laurens Hammond was an

Figure 16: Adding 'Hammondesque' vibrato.



engineer, not a musician, and reputedly tone-deaf. Yet he had very strong views regarding the tone that he wanted from his organs, and gave explicit instructions to his factory and service centres that the amplifiers were to be adjusted so that there was no overdrive or distortion. Nowadays, we think that Hammond was wrong, and overdrive and distortion have become invaluable in all forms of non-classical music. To be fair to Mr Hammond, it was only in the 1950s that keyboard players and guitarists started to experiment with overdrive seriously, and it took another decade for distortion to emerge as a fundamental building block of modern popular music.

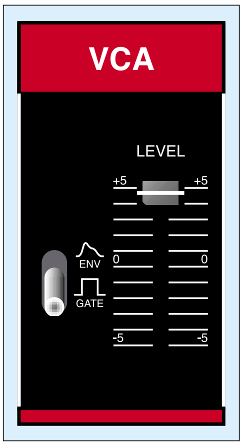


Figure 17: Adding distortion.

Nowadays, many synths feature digital overdrive/distortion effects, but the Juno predates such enhancements. Nonetheless, all is not lost, because with the high internal signal levels generated by the DCO, the sub-oscillator *and* the self-oscillating filter, it is easy to overdrive the Juno's VCA by raising its Level toward +5 (see Figure 17, left). The result can be anything from a mild distortion to a full-throated crackle. It's not the same as the warm burr of a 30-year-old valve on the edge of break-up, but produces some very useable results, plus an unexpected side-benefit. A Hammond exhibits mild compression when you add notes to a chord and, coincidentally, an overdriven VCA exhibits exactly the same quality when you exceed the limit of its abilities to amplify and drive it into clipping distortion.

Unfortunately, you can't employ this trick on many synths, because the majority are factory-calibrated to stop you from clipping the signal. This is understandable; for most sounds, the results would be inappropriate and unpleasant. Still, it would be nice if the option existed, as on the Juno.

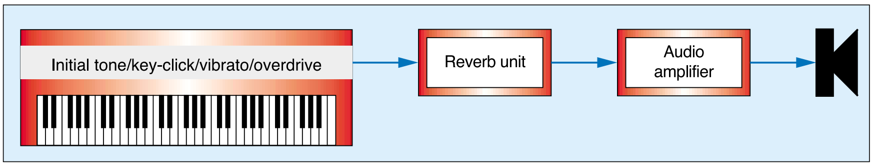


Figure 18: The conventional use of a reverb unit.

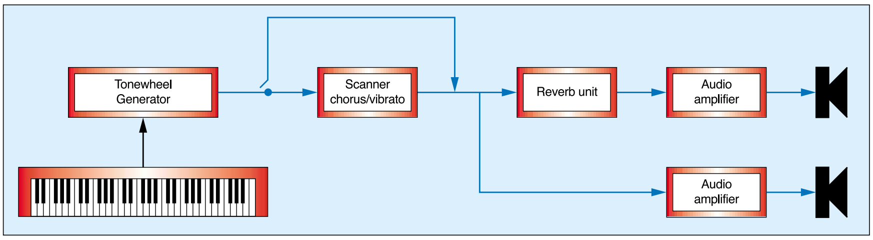


Figure 19: A simplified schematic of the Hammond A100.

## Hammond Effects — Reverb

In some low-cost Hammonds, the next element in the signal path is a spring reverb unit. You would think that it would be a doddle to imitate this... why not just plug a suitable spring reverb or digital imitation between the Juno and the amplifier/speaker system, as shown in Figure 18? However, this is not quite right, because the overdrive generated by the overdriven VCA occurs before the reverb unit, and this is the opposite of what happens in the Hammond. Nonetheless, many modern reverb units offer suitable effects, provided that you disable all the extra stuff that they tend to offer.

Things become more complex when you consider the A100, which has a separate amplifier and speaker to handle the output from the reverb unit (see Figure 19). However, this is easily recreated, because many digital reverb units allow you to send a treated signal to one channel while directing the original to another. This means that I can draw Figure 20, with a modified Juno patch providing optional vibrato and overdrive, played through two channels; one clean, the other reverberated.

So... how does it all sound? The truth is, not great. I don't like the vibrato effect, we've been unable to synthesize percussion or chorus, and while the distortion effect is quite pleasing, sticking a digital reverb after a patch doesn't count as 'real' synthesis. Sure, we've learned a great deal simply by attempting to recreate the Hammond effects, but it would have been nice to achieve something more satisfying. Fortunately, this isn't the end of the story, because I've left the most important — and by far the best — organ effect out of this discussion. I'm referring, of course, to that generated by the rotary speaker or 'Leslie' attached to almost all A-, B- and C-series Hammonds. So, next month, we're going to wrap up our synthesis of the Hammond organ by getting ourselves into a bit of a spin. SOS

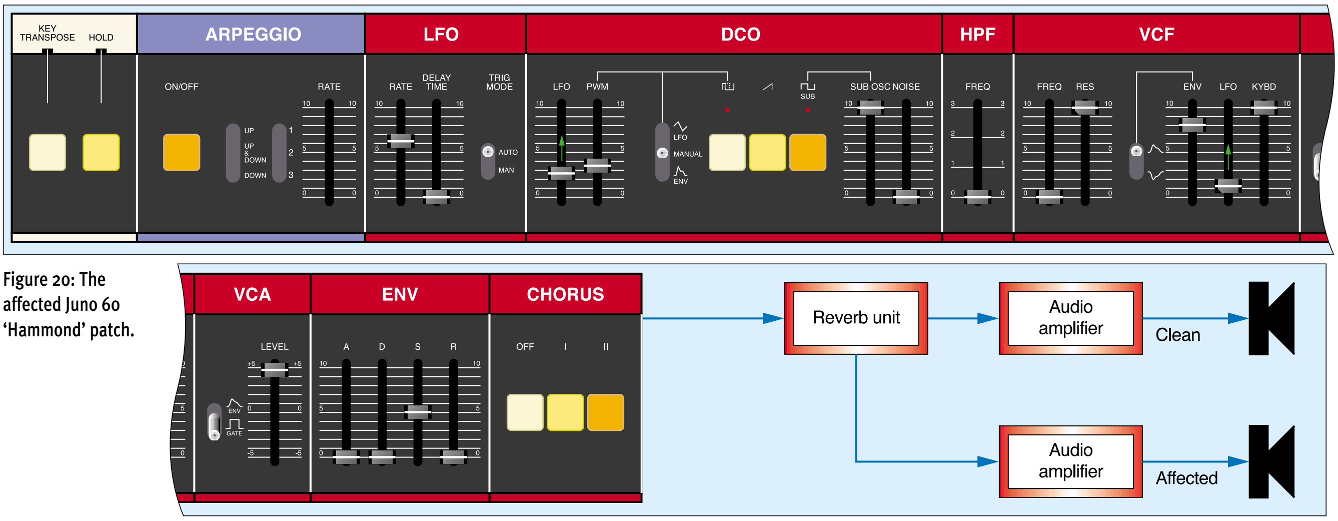


Figure 20: The affected Juno 60 'Hammond' patch.

# Synth Secrets

Photo: Richard Ecclestone

## Synthesizing The Rest Of The Hammond Organ — Part Two

**As with so much surrounding the Hammond organ, there's much more to the Leslie rotary speaker than meets the eye, and synthesizing its effects involves considerably more than just adding vibrato, as we find out this month...**

Gordon Reid

For three months, we've been investigating the sound of the Hammond organ, spending two of those months recreating the sound of the tonewheel generator, and a third attempting somewhat less successfully to emulate the onboard effects provided by 'the real thing'. But, as we all know, the classic jazz/rock/pop organ sound is as much a consequence of a rotary speaker as it is of the organ itself. So, in this fourth article dedicated to understanding and synthesizing the organ, we're going to concentrate on the physics and sound of the 'Leslie' speaker.

### A Brief Description

What we now recognise as the rotary speaker did not materialise as a fully developed concept overnight. In an effort to animate the sound of the Hammond electromechanical organ, Don Leslie had been experimenting with all manner of systems before he alighted upon what he called a 'Vibratone Speaker', but which is what we now call the classic, twin-rotor 'Leslie'. One of his early experiments was a monstrosity with 14 speakers mounted inside a rotating drum. Fortunately, while paring his ideas down to a manageable size and complexity, Leslie found that just two speakers (mounted inside a cabinet, as shown above) generated the most pleasing sound. One of these was a treble unit that produced the frequencies above 800Hz, and which played upward into what looks like two rotating horns (although one of these is a dummy, provided only to stop the whole assembly from shaking itself to bits). The second was a bass speaker reproducing frequencies below 800Hz, played downwards into the single rotating 'rotor' (see Figure 1, below).

Leslie provided each speaker assembly with two rotation speeds — one slow and one fast — that he called 'chorale' and 'tremolo', respectively. The rotor's chorale speed was different from the horn's, as was its tremolo speed, and the transition rates between slow and fast (and *vice versa*) were different for the two assemblies. The result of playing a sound through this device was, therefore, a complex effect that combined pitch modulation, amplitude modulation, tone modulation, and reverberation (ie. the effect of the enclosed cabinet), with the high frequencies and low frequencies swirling around independently to create a very pleasing and expansive sound. What's more, Leslie found that, by placing the cabinet close to a wall or to the corner of a room, he could use the additional reflections from those surfaces to obtain a stereo field from a single, monophonic source.

Nowadays, there are dozens of models of Leslie speaker, and numerous physical as well as electronic imitations. Some Leslies have single rotors, but the sound produced by these is inferior to that offered by the larger, dual-rotor models characterised in Figure 1. Others are built into their host organs; a common example is the single-rotor unit found within the Hammond T500-series organs from the mid-1970s. And, of course, no modern organ emulation could be considered complete without the inclusion of

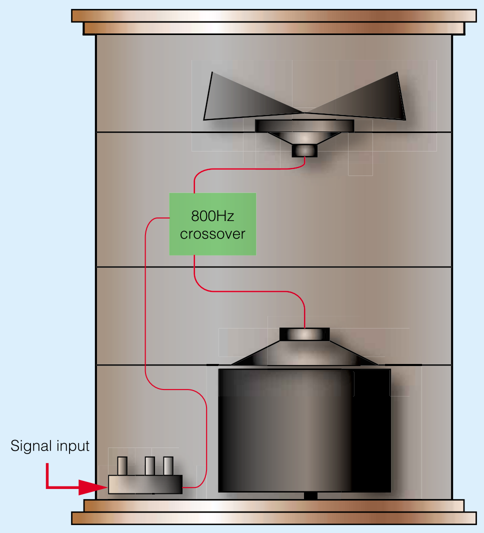


Figure 1: Don Leslie's twin-rotor speaker cabinet.

a digital recreation of the Leslie speaker effect.

But how many of us fully understand how the Leslie creates its instantly recognisable sound? Ask many keyboard players what makes the device so special, and they will usually offer an answer that includes the words 'Doppler effect'. They are right to do so — the Doppler effect can explain a significant element of the sound generated by the Leslie. However, if you then ask what the Doppler effect is, you will often be met with a blank stare. What's more, this is only *part* of the story, as we shall now see...

## The Doppler Effect

Christian Johann Doppler was a scientist who died as long ago as 1853, yet he explained something that we all experience; the fact that, when something emitting a sound is moving towards us, we perceive a higher pitch than when the same sound is moving away from us. To understand why this is the case, we'll start by following Doppler's line of thinking, and consider an archetypically 19th-century mode of transport: a ship.

Imagine, if you will, a docked liner floating in a harbour with the tide coming in. If this is facing into the direction of the tide, the waves will be breaking against the bow once every few seconds. Let's say — for the sake of argument — that the peak of each successive wave is reaching the bow five seconds after the previous one, as shown in Figure 2 (above).

Now let's imagine that the ship is surging through the seas, with the bow cutting through the waves like... well, like a million tonnes of cruise liner cutting through the waves. If the direction of the tide is unchanged and the ship is still pointing in the same direction, the time between wave crests will be much reduced — say, to one crest every three seconds (see Figure 3 above).

Finally, let's imagine that the ship has swung right around and is heading back the way it came. As you would expect, the wave crests are now reaching the stern at a slower rate than before — say, once every seven seconds, as shown in Figure 4 (above). The frequency of the waves hasn't changed, but by moving in the same direction as the waves, or in the opposite direction to them, we have demonstrated that the frequency at which we observe a waveform is relative to our motion with respect to it. This is the effect that Doppler not only observed but also quantified, so history has seen fit to call it the Doppler effect.

Now, you don't need to be standing on an ocean liner to experience the Doppler effect. An equivalent shift in pitch occurs when you are stationary and something emitting a wave — say, the siren mounted on top of an ambulance — is moving toward you or away from you. In the first instance, the vehicle has moved a little closer to you as its siren emits each subsequent peak in the waveform, so the wavelength is shorter than it would otherwise be, and therefore higher in pitch (see Figure 5, above). The reverse occurs when the ambulance is moving away from you, with the siren a little further away each time it emits a peak. In this case, the waveform is lengthened, and the pitch is lowered (see Figure 6, above).

Matters are complicated slightly when the ambulance is not travelling *directly* towards or away from you. Consider the case where you are standing on the pavement as it goes by. At the exact moment it passes you, it will neither be approaching or receding, and you will hear the true pitch emitted by its siren. So it should be intuitively clear that the change in ▶

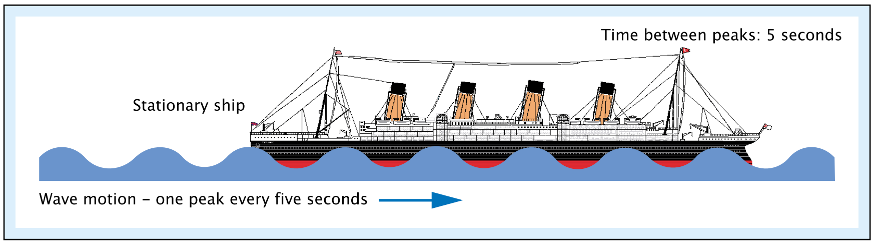

Time between peaks: 5 seconds
Stationary ship
Wave motion – one peak every five seconds

Figure 2: Observing the waves from a stationary ship.

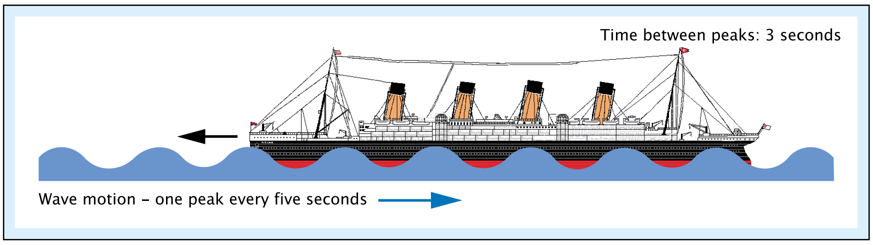

Time between peaks: 3 seconds
Wave motion – one peak every five seconds

Figure 3: Moving into the waves increases the perceived frequency.

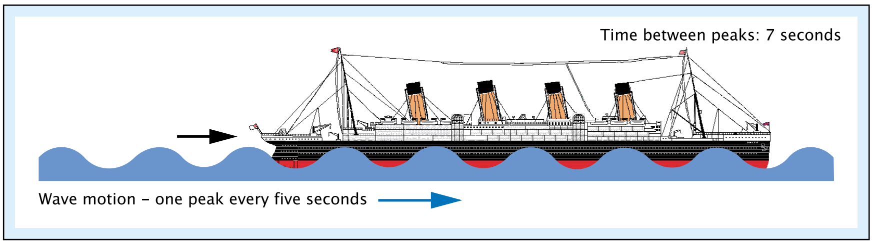

Time between peaks: 7 seconds
Wave motion – one peak every five seconds

Figure 4: Running *from* the waves reduces the perceived frequency.

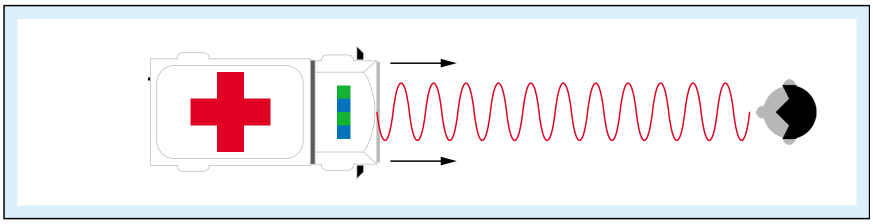


Figure 5: The pitch of an approaching ambulance siren is higher than if stationary.

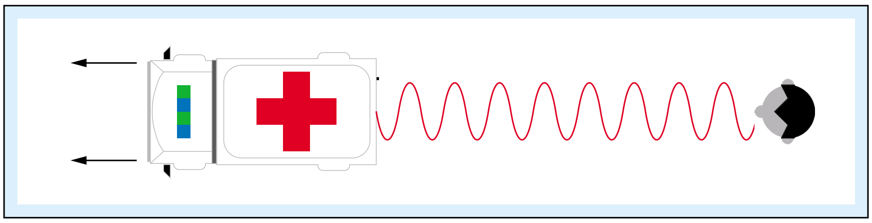


Figure 6: The pitch of a receding ambulance siren is lower than if stationary.

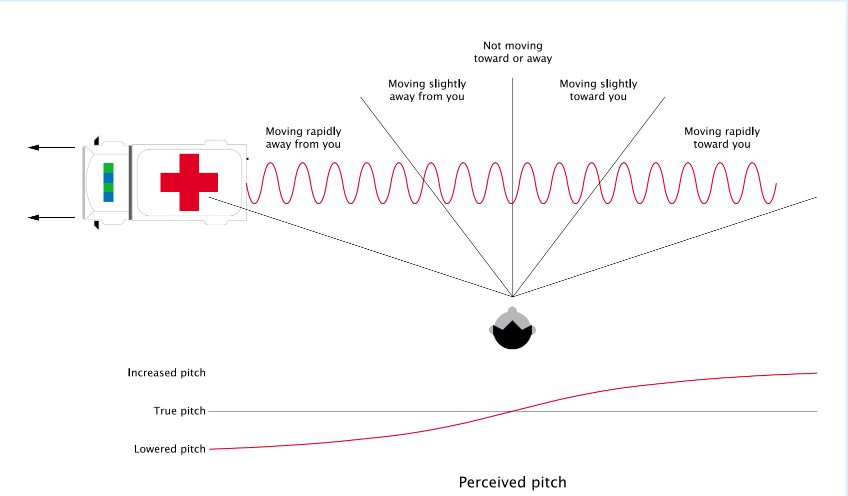


Figure 7: How the pitch changes as the ambulance passes.

pitch describes some sort of curve as the ambulance passes. This is the case illustrated in Figure 7 (above).

## Analysing The Rotary Speaker Cabinet

Everything I've described so far is relatively straightforward, but it doesn't explain the sound of the rotary speaker cabinet, so please bear with me while I extend the 'ambulance' analogy somewhat further, whereupon all will become clear.

Instead of considering the ambulance siren moving toward or away from you in a straight line as shown in the diagram above, imagine that the vehicle is stuck on a roundabout, forever circling as it fails to find an exit. Clearly, the siren's pitch will appear to be raised when the vehicle is moving towards an observer on the roundabout, and lowered when it is moving away. There will also be two instances — when the ambulance is neither moving toward nor away from the observer — when the pitch is heard unaltered by the Doppler Effect. Without going into the trigonometry of the situation (which would involve a little mathematics and no doubt elicit yelps of pain on the *SOS* Readers' Forum) I can tell you that — provided that the speed of the ambulance is constant — the change in perceived pitch is described by a sine wave (see Figure 8, below left).

Now, if we replace the ambulance and siren in Figure 8 with the aperture of a horn speaker, we can see that the analogy explains the first element in the rotary speaker effect. As the horn aperture moves toward you, the perceived pitch of the sound it is radiating is raised; when the aperture moves away from you, the perceived pitch is lowered; and the nature of the pitch-shift is again described by a sine wave (see Figure 9, below left).

Many writers have stopped at this point, claiming that this explains the sound of the Leslie speaker. But this can't be right; all we've described so far is a mechanical method for generating a simple vibrato. So let's think about the situation a little more deeply...

Looking again at Figure 9, it should be obvious that the pitch of the note is not the only thing affected by the rotation of the horn. In particular, the perceived sound is going to be much louder when the horn is pointing towards you than it is when the horn is pointing away from you. What's more, it's going to have some intermediate loudness when the horn is pointing sideways. It doesn't take a genius to realise that the perceived loudness curve is also going to be a sine wave, or something very similar. This means that the motion of the horn aperture is generating not one, but *two* sonic effects. The first is vibrato, with its peak occurring when the horn is moving fastest toward you (when it is pointing to the right in Figure 9), while the second is an amplitude modulation — or tremolo — with its peak occurring when the horn is pointing straight at you. This means that the vibrato is 90 degrees out of phase with the tremolo (see Figure 10, on the next page). If you're not sure what '90 degrees out of phase' means, take a look at Part 4 of this series, way back in *SOS* August 1999, or check out www.soundonsound.com/sos/aug99/articles/synthsecrets.htm.

> **"... the motion of the horn aperture is generating not one, but *two* sonic effects."**

Although this may seem a little complex, it's very simple to synthesize if we use an oscillator as the sound source. It requires just four modules to generate the effect; the VCO generates the initial sound, a sine wave LFO emulates the rotary motion of the horn, and a delay line connected to a VCA

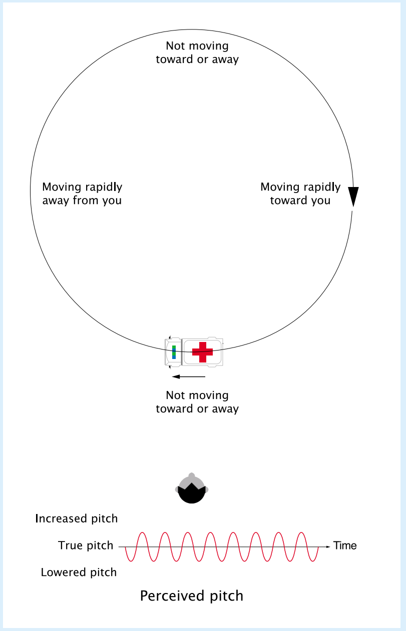


Figure 8: The observed pitch when the ambulance is stuck on a roundabout.

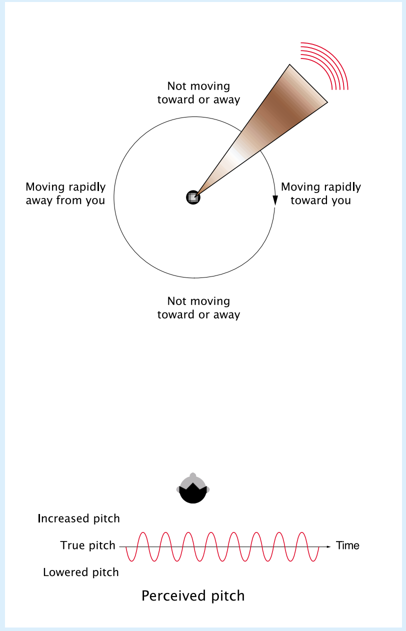


Figure 9: The perceived pitch shift of a rotating horn speaker.

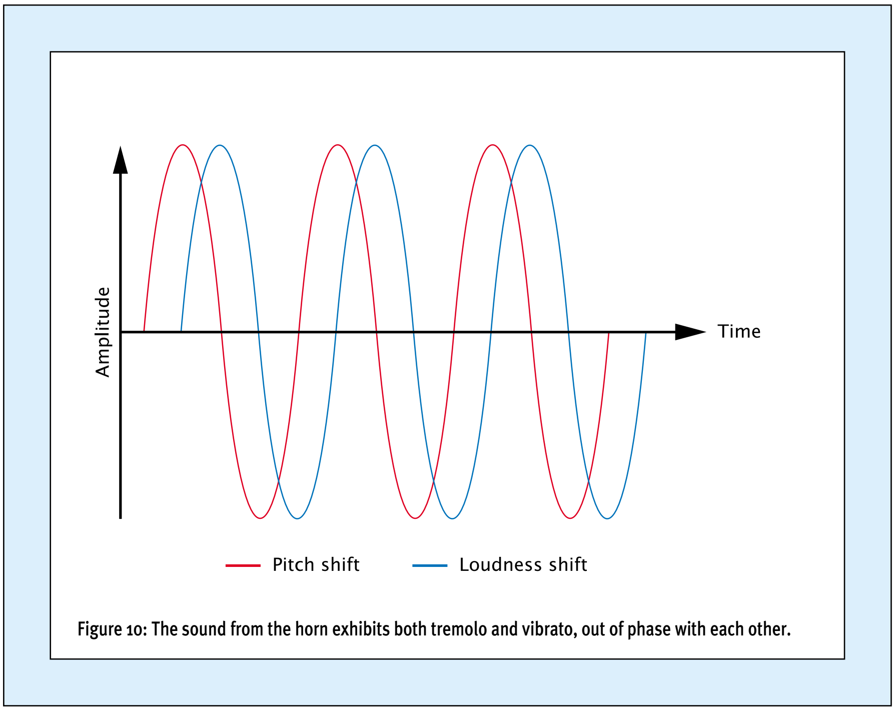


Figure 10: The sound from the horn exhibits both tremolo and vibrato, out of phase with each other.

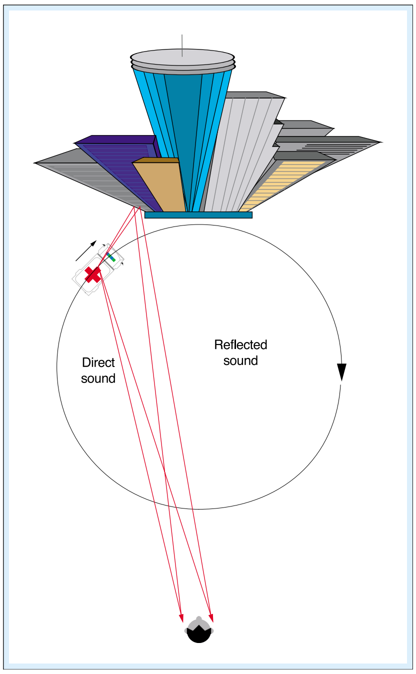


Figure 13: Reflecting the sound off an office block.

introduces the phase-shift between the frequency modulation and the amplitude modulation. Hooked together as in Figure 11 (below), these would generate a waveform exhibiting both vibrato and tremolo, but with the two 90 degrees (or some other desired amount) out of phase with one another.

Unfortunately, this is still not a complete description of the effects imparted by the rotating horn, because the *tone* of the sound will also change throughout the cycle. The manner in which it does so is not intuitively clear. It is certainly not straightforward, because to understand this we would need to analyse such nasties as the backward projection from the horn driver, introduce some fluid dynamics to determine how sound is propagated 'backwards' through the air, and look into the refractive edge effects of the horn itself. Trust me... these are matters best left alone unless you fancy studying acoustics for a PhD. Nonetheless, we can say with some confidence that, whatever other changes take place, the sound will be brighter when the horn is pointing towards us, and duller when it points away. This suggests that the tonal modulation lies *in phase* with the loudness modulation, and that we can synthesize this — to a first approximation — by adding a low-pass filter modulated by the delayed LFO signal that is driving the VCA in Figure 11. The result looks like Figure 12 (see below).

## Cabinet Reflections

However, even this is far from a description of a rotary speaker, because neither the horn nor the rotor in Figure 1 are rotating in free space. So let me return for a moment to the analogy of an ambulance stuck on a roundabout. Imagine that, on the opposite side of the roundabout, there's a large, reflective surface of some sort... say, a large office block (see Figure 13, above right). This will reflect back some of the siren's sound that would otherwise travel forever away from you, and you'll hear this mixed in with the direct sound.

If you consider what is happening to the *reflected* signal, you'll appreciate that — due to the Doppler effect — the pitch of the sound is at its highest when the ambulance is moving toward the office block, and its loudness and brightness are greatest when the ambulance is alongside the office block (ie. at its point furthest from you). If we ignore, for a moment, the finite speed of sound, this means that the pitch, loudness, and brightness modulations of the reflected signal are 180 degrees out of phase with those of the direct signal. However, we *can't* ignore the finite speed of sound, so what we hear is delayed by an additional amount proportional to the greater distance that the reflected sound must travel. This means that the phase change of the effects is not 180 degrees, but some other amount, as illustrated in Figure 14 (on the next page).

But this still isn't the end of the story, because we have not yet taken into account the additional changes in tone that occur as the sound is reflected off the surface, and as it is absorbed by the air. Experience teaches us that, if the original signal occupies a broad band of frequencies, the direct sound will be brighter than the reflected

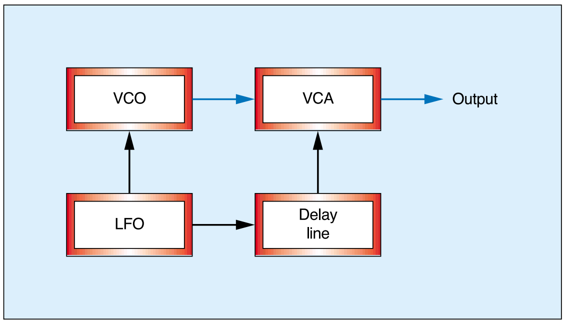


Figure 11: Recreating the phase and modulation characteristics of Figure 9.

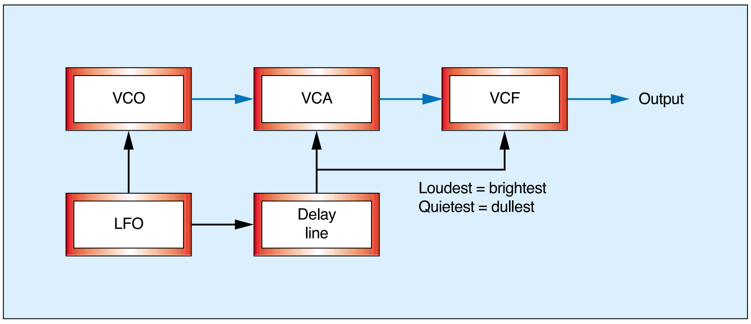


Figure 12: Adding tone modulation to Figure 11.

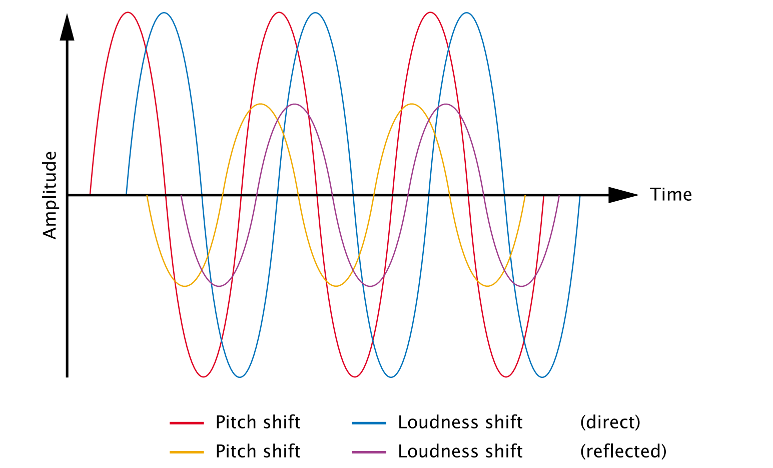


Figure 14: Considering the phase and amplitude of the reflected sound.

▶ sound. I think that it's safe to adopt this as an accepted — if unproven — part of our analysis because firstly, the reflected sound travels further, so more high frequencies are absorbed by the air, and secondly, the building will reflect lower frequencies better than others, thus imparting a duller tone to the sound. In addition to this, the delayed signal is going to interfere with the direct signal, resulting in constructive and destructive interference, which in turn will result in comb filtering of the sound that you hear.

We can synthesize Figure 13 in block-diagram terms as shown in Figure 15 (below). In this, I have added a second set of delay lines, filters and VCAs to Figure 12, thus creating a second signal path that provides the delayed, lower-amplitude 'reflected' signal. In an attempt to be as accurate as possible, I have also added a gentle low-pass filter in the second signal path, which emulates the additional loss of high frequencies in the reflected signal.

Clearly, placing just one reflector in the system complicates matters hugely, but that is as nothing compared to the complexities of a real rotary speaker cabinet which (ignoring the top and bottom) has four sides. If I may use my analogy one last time, this would be like placing three more office blocks on the remaining sides of the roundabout, such that the sound source is surrounded by reflective surfaces (see Figure 16, on the next page). It's pretty clear that the interactions between the enormous number of signal paths thus created are going to become very complex, very quickly.

## Synthesizing Some More Reflections

Contrived though this analogy may seem, it's a surprisingly good description of the physics of a rotating speaker. The only difference in the geometry is that, instead of escaping through gaps in office developments, the sound of a rotary speaker escapes through holes cut into the sides of the cabinet. This means that we can use this model of pitch-shifts, amplitude-shifts, tone modulations, and reflections to develop ways of synthesizing the Leslie itself. But be warned... the solutions are far from trivial. Indeed, the fact that there was never a convincing 'analogue' Leslie effect is a dead giveaway. Consider this: Figure 15 is already starting to look rather complex, but we now have to imagine what happens when we add all four walls of a Leslie cabinet (ie. the multiple reflective surfaces in Figure 16) and try to synthesize all of the signal paths thus generated.

The result is completely impractical, both in terms of the number of analogue synth modules required, and in the amount of paper that we would need to represent them. Figure 17 (also on the next page) shows the modules required to model just three reflections, and this is already becoming a nightmare. When you consider that wherever you may stand in space, the sound you hear coming from a Leslie comprises many thousands of such paths, you can appreciate the size of the problem. What's more, sophisticated though Figure 17

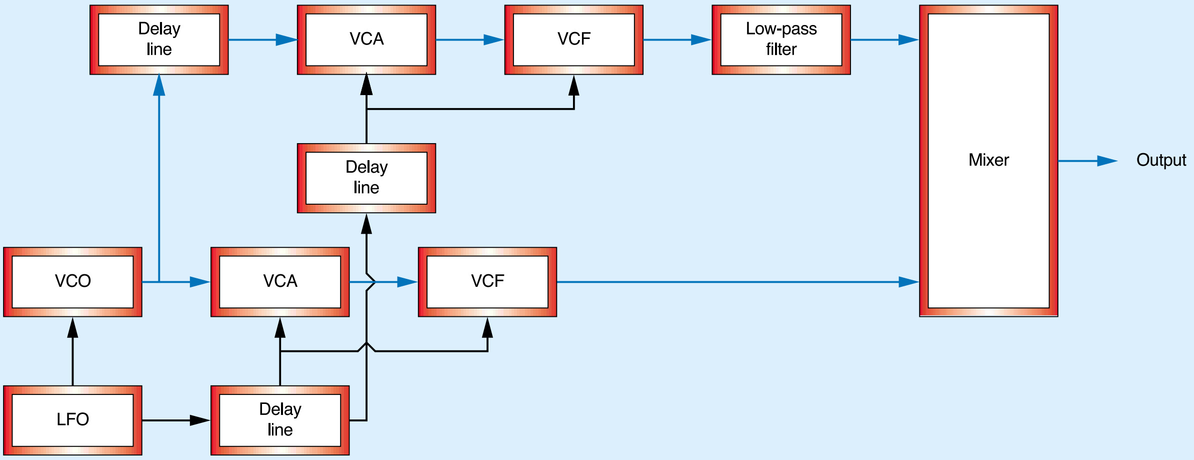


Figure 15: Synthesizing the direct 'rotary' sound plus one reflection.

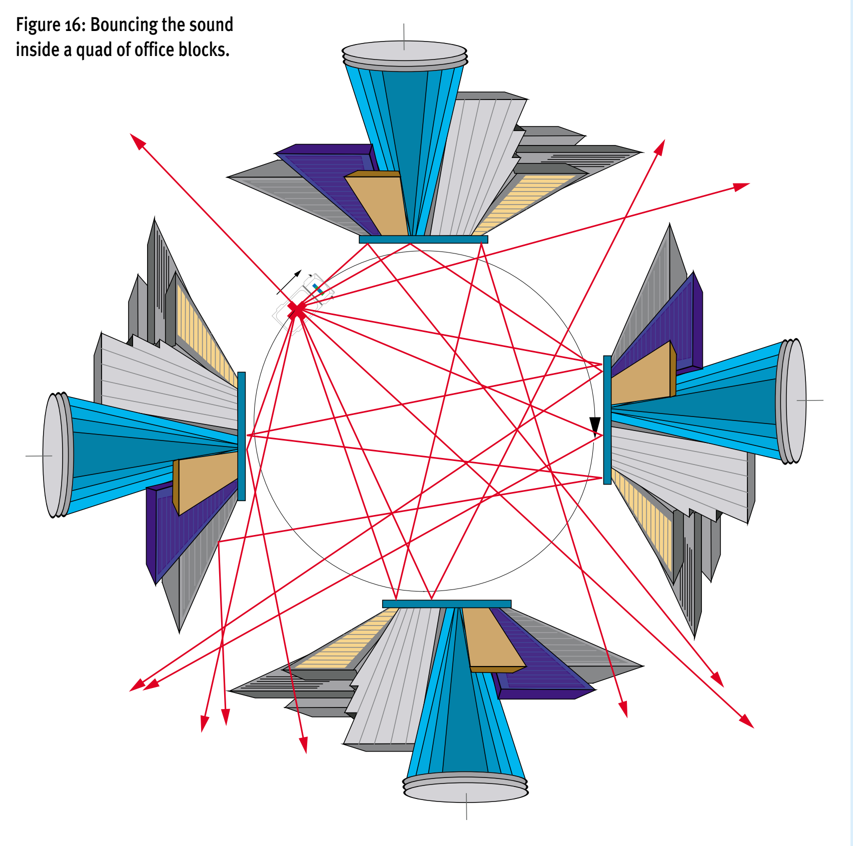
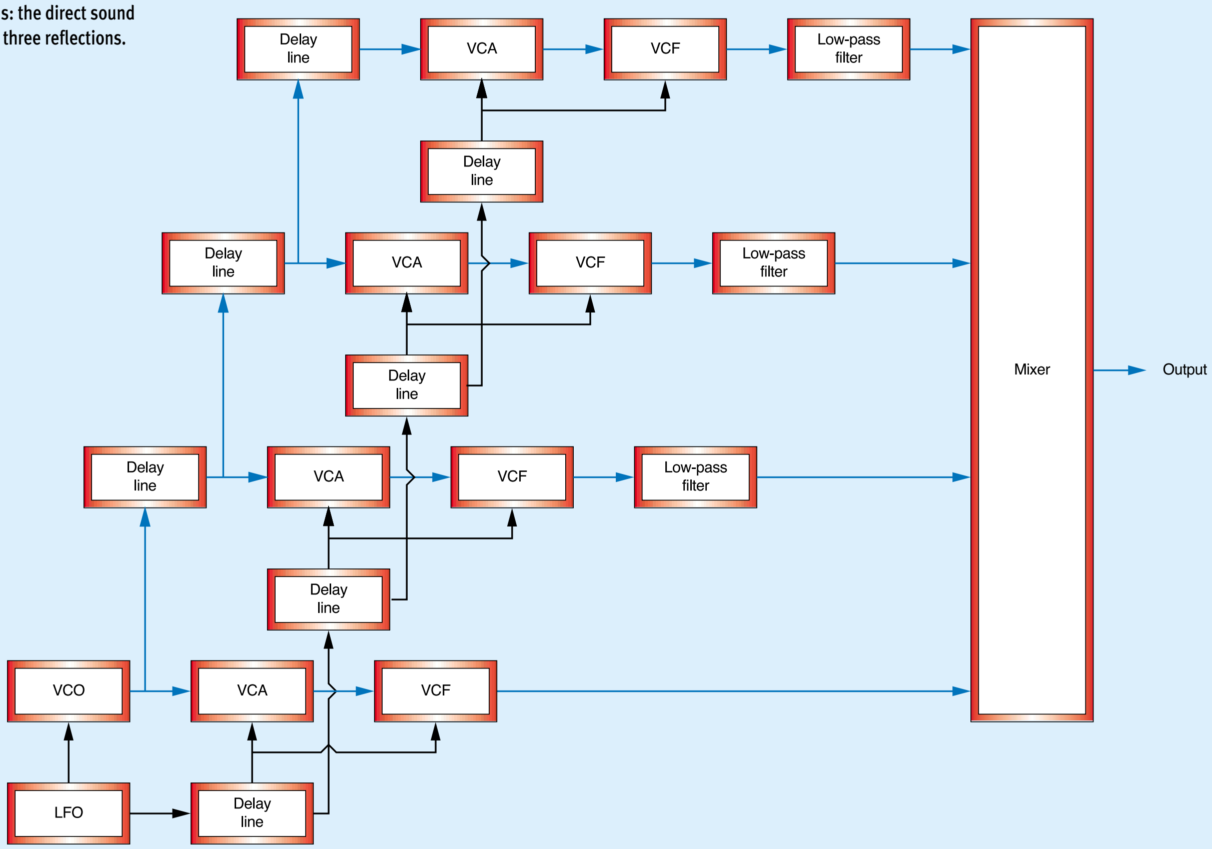


Figure 16: Bouncing the sound inside a quad of office blocks.

appears to be, it is in fact rather inelegant, with all of its individual VCFs, multiple delay lines, and all the signal paths' low-pass filters set to different cutoff frequencies. However, there's a bigger problem. In Figures 12, 15 *and* 17, we have been modulating the pitch of the sound source (the VCO) directly, rather than modulating any possible sound that we might want to affect. While this might be satisfactory for some organ sounds (for which we can modulate multiple VCOs to create a tonewheel/Leslie effect) it is unsuitable for synthesizing the complexities of, say, a guitar or human voice played through a Leslie speaker. To do this, we need to be able to input an external signal and effect this.

Fortunately, there is a way, and we can use a particular analogue device to modulate the pitch of any input signal. This will then give us a creditable chance of imitating the effect imparted upon any sound source played through a Leslie cabinet. Figure 17 even contains some of the information that we need to do this. Unfortunately, we have run out of space for this month, so next time we'll begin looking at some practical methods for synthesizing rotary speakers. Until then... SOS

Figure 17: Four signal paths: the direct sound plus three reflections.



# Synth Secrets

Photo: Richard Ecclestone

## Synthesizing The Rest Of The Hammond Organ — Part Three

Gordon Reid

**We conclude our analysis of the fabulously complex beast that is the Leslie rotary speaker.**

Last month, we analysed the nature of the 'Leslie' rotating speaker system, and I showed how any signal played through such a device is subjected to frequency modulation, amplitude modulation, tone modulation, and reverberation. I also showed how — in principle — we could use an LFO connected to the pitch modulation input of an oscillator, plus various filters, amplifiers and delay lines, to emulate this effect. But the weak link was the oscillator and LFO. It's all very well modulating the pitch of a signal produced electronically in this fashion, but this method gives us no clue as to how we could modulate *any* signal, such as that produced by an organ, a guitar, or a human voice. If we fail to find a way to do this, we cannot properly synthesize the rotary speaker. On the other hand, since such effects exist, and existed long before the development of modern digital units, it can't be that hard... can it? After all, many players used analogue rotary speaker effects in the 1970s, even though they weren't particularly convincing.

Of course, these days, there are plenty of available digital rotary speaker simulators, but as with previous instalments of this series, I'm going to describe the process using analogue principles, as it's easier that way to relate the constituent parts to conventional synthesizer components, and understand how everything works.

Let's start by returning to what this series was examining way back in *SOS* August 2000 (see www.soundonsound.com/sos/aug00/articles/synthsec.htm). That month, I showed how the concepts behind Sample and Hold (or S&H) synth modules are related to those behind analogue-to-digital converters, and thus to all of digital audio. Today, I find myself at the same starting point, and, although it may not be obvious how discussions of S&H circuits and Leslie speakers should be so closely linked, I'll ask that you bear with me because — as always — all should soon become clear.

### A Quick Recap

To understand S&H and how it leads to the technology of modulated effects, I'm going to review some of the ground that we covered back in 2000, starting with Figure 1 (below), which I've copied from the previous article. As you can see, this is a remarkably simple circuit, comprising just two components: a capacitor and a switch.

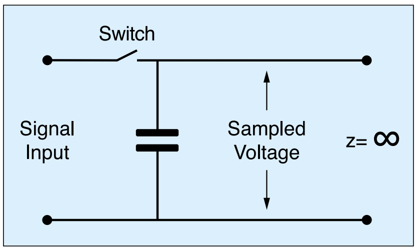


Figure 1: The simplest representation of a S&H circuit.

There's nothing stopping us from presenting an audio signal, an LFO, an envelope, or anything else to the input in Figure 1, as I did back in 2000. However, this month, I'm going to concentrate on presenting audio signals, starting with a simple sine wave.

Imagine that, just for an instant, the switch in the diagram closes. If the capacitor can react quickly enough, it then charges up (or discharges down) to the voltage at the input, thus 'sampling' that voltage. Then, once the switch has opened again, the voltage across the capacitor cannot change. This is because, on the left-hand side, there is no circuit and, on the right-hand side, the impedance — which is represented by the mathematical symbol 'z' — is infinite (which means that no current can flow). However, although no current flows, you can still measure the voltage across the output.

That's all there is to it... when the switch is closed, the capacitor 'samples' the input voltage. When the switch is open, the capacitor 'holds' that voltage, allowing other circuits to respond to it as appropriate.

Now, if you were limited to closing the switch in Figure 1 manually, this S&H circuit would not be of much use. So synthesizers have electronic switches that respond to another module that is capable of opening and closing it at high speeds. This 'other module' is a Clock Generator, which provides a stream of pulses that trigger the switch in Figure 1 (see Figure 2). In other words, when the pulse is on, the S&H circuit samples, and when the pulse is off, the S&H holds.

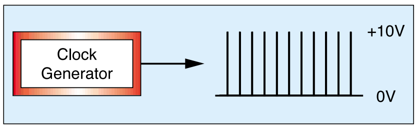


Figure 2: The output from a Clock Generator.

Given these two modules, we can devise a simple circuit that incorporates the clock and the S&H circuit, as shown in Figure 3 (on next page). In this case, this shows

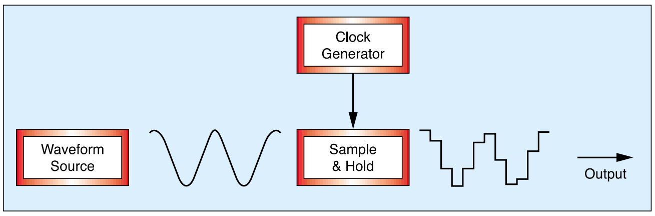


**Figure 3: A simple example of S&H.**

something akin to a sine wave presented to the signal input of the S&H module. At the same time, the clock provides a stream of pulses that it presents to the S&H's trigger input. The output produced by the S&H circuit is then the 'blocky' waveform shown.

Figure 4 (below) explains the nature of the output. Each time the S&H module receives a trigger, it measures (or 'samples') the voltage of the input signal (shown in red). It then holds this voltage (the blue line) until it receives the next trigger, at which point it repeats the operation. It 'samples' and then 'holds', just as I've described.

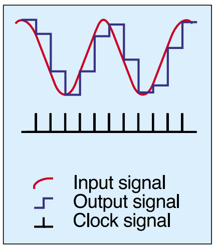


**Figure 4: Explaining S&H.**

As I suggested last time, this result would not be very interesting if a sine wave was the only signal you could present to the module's input. Fortunately, the input signal can be anything: a synthesized audio waveform, an external signal such as the output from a turntable or  CD player, or even the 'live' sound of an instrument being played. And this is where we begin to diverge from my previous discussion. Whereas traditional synth S&H effects are derived mostly from using a random 'noise' signal as the input, and directing the output to the control inputs of other synthesizer modules, we are now going to concentrate on affecting the actual sound of an instrument being played. But before we do so, we have to convert the S&H circuit into a delay line...

## The Bucket Brigade Device

Let's place two S&H circuits in series, as shown in Figure 5 (at bottom of page). The white triangles are 'buffer amplifiers'. They provide the infinite impedances mentioned above, but do not affect the signal in any other way. Consider what happens when Switch 1 and Switch 2 are open, and then Switch 1 closes for a moment, before opening again. When Switch 1 closes, a single sample is taken and held by the first S&H circuit.

Now imagine that this sequence of events repeats, but that this time it is Switch 2 which closes for a moment, and then re-opens. When Switch 2 closes, the second S&H circuit takes the sample held in the first as its input, samples it, and holds it. In other words, the sample is passed down the line!

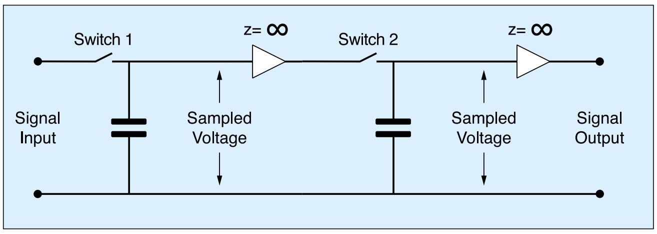


**Figure 5: Two S&H circuits in series.**

It takes little imagination to realise that we can extend this idea, adding as many elements to this circuit as we like. Take Figure 6 (below) as an example. This has eight S&H stages. If we open and close all the red switches simultaneously, and all the green switches simultaneously, closing the green when the red are open and *vice versa*, we can take a continuous stream of audio samples at the input and pass them down the line to the output. If the clock rate is, for the sake of argument, 44,100 pulses per second (the standard CD sampling rate), the length of the delay line is 8/44,100 seconds, which is somewhat less than 0.2 milliseconds... far too short to be of use. But if we extend this to 2048 stages, the length of the line is more than 46ms, which is long enough to create a range of common audio effects. What we have here is a sampler — one that is entirely analogue, too.

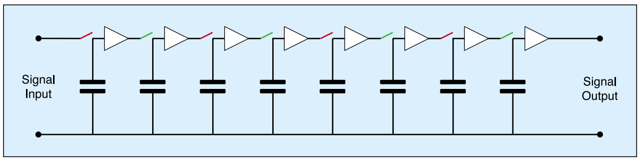


**Figure 6: An eight-stage delay line.**

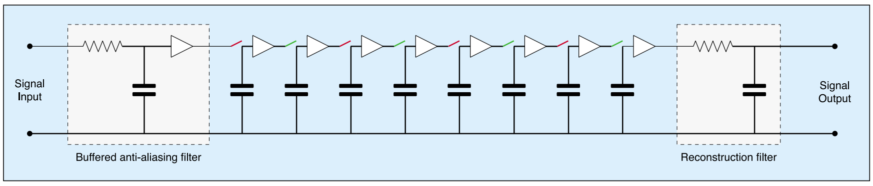


**Figure 7: Adding an anti-alias filter and a reconstruction filter to the delay line.**

Before moving on, we need to eliminate two problems encountered when sampling and reconstructing a continuous waveform. Just as when sampling digitally, all that stuff about keeping the maximum frequency at less than the sampling rate holds true here, too (for more on this, look back at part 17 of this series, in *SOS* September 2000, or at www.soundonsound.com/sos/sep00/articles/synthsec.htm).

Because of this, we need to ensure that the highest frequency presented to the delay line is less than half the sampling frequency. In this case, the sampling frequency is half of the clock frequency, because, as illustrated in Figure 5, a new sample is taken every two trigger pulses. Anyway, to ensure that the input is suitably band-limited, we need to add a low-pass filter before the signal input. Secondly, we want to eliminate the 'blockiness' from the output waveform shown in Figure 3, and we do so by smoothing the output using a second low-pass filter.

Putting all of this together, we now have a circuit description for an analogue 'bucket-brigade device' (or BBD) delay line, so-called because its operation is analogous to handing buckets of water, each filled to a different depth, along a line of people (see Figure 7, above).

By the way, the low-pass filters I have drawn — simple 1-pole devices — are much less powerful than one would normally use for these purposes, so please treat them as representative rather than an exact circuit description. The first of these is called an 'anti-aliasing filter' because it removes high frequencies that lead to aliasing. The second is known as a 'reconstruction filter' because it reconstructs the smooth waveform from the 'blocky' one at the output.

▶

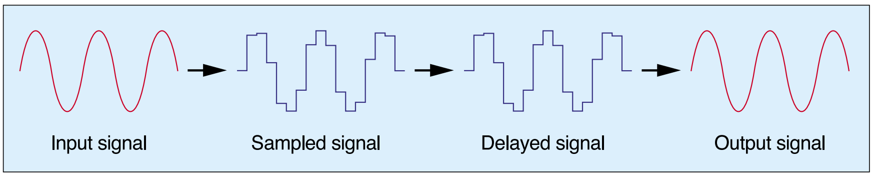


Figure 8: Sampling, delaying and reconstructing the waveform.

## Clock Modulation & Waveshaping

To properly emulate a Leslie, you need not only to delay the signal passing through it, but also to modulate its frequency. To do this, let's return to the clock that's opening and closing the switches within the delay line. If the speed at which the clock is running remains constant, the signal will be sampled steadily, with each sample passed down the line at a constant speed and then read out with the temporal gaps between the output samples equal to the gaps at the input. If the reconstruction filter does its job correctly, the output waveform should then be identical to the input waveform (see Figure 8, above).

But what happens if we modulate the clock so that adjacent samples measured at one rate are presented to the reconstruction filter at a different rate? For the purpose of this discussion, you can think of the delay line as a tape recorder, with a record head at one end and a playback head at the other, and an infinitely long strip of tape running past them. If the speed of the tape is, say, 15ips when a middle 'C' (C3) is recorded at the start, but just 7.5ips when that part of the tape passes the playback head, the note will be replayed as C2, an octave lower. Conversely, if the tape speeds up to 30ips, the note will be raised an octave, and reappear as C4.

Of course, we certainly don't have to restrict ourselves only to increasing or decreasing the speed of the tape. If we could *modulate* the tape speed in some fashion, we could generate pitch modulation, or 'vibrato'. Now, let's return to the delay line, and modulate the clock, so that the relationships between samples are changed slightly...

Figure 9 (left) shows approximately 24 cycles of a sine wave that, for the sake of argument, I have presented to the input of our delay line. I shall now modulate the clock frequency to obtain Figure 10 (left), which shows that I have increased the wavelength of some cycles, thus lowering the frequency, and decreased the wavelength of others, thus *raising* the frequency. It should be obvious from this somewhat exaggerated example that this is an extreme example of pitch modulation.

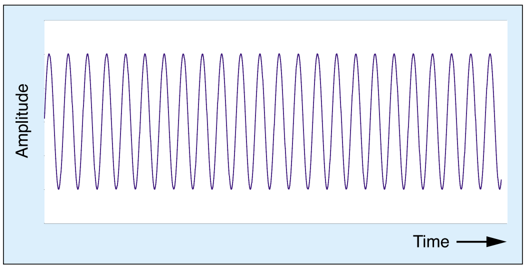


Figure 9: Presenting an audio sine wave to the delay line's input.

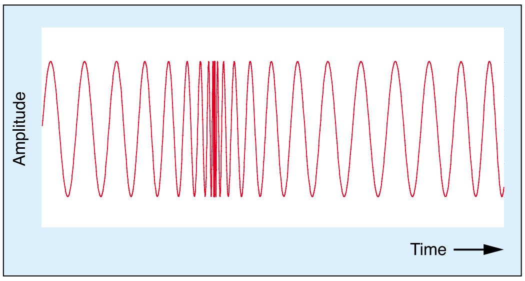


Figure 10: Modulating the output clock to generate pitch modulation.

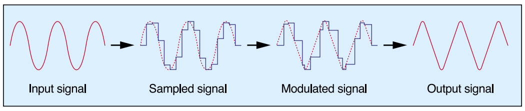


Figure 11: Waveshaping by modifying the clock frequency.

The great thing about this method of generating vibrato is that, unlike presenting a pitch CV to the modulation input of an oscillator, it allows us to modulate *any* input signal. It's also interesting to note that, if we increase the speed of the clock modulation, the output waveform will be altered in a more radical fashion. For example, Figure 11 shows how the samples in Figure 8 can be re-timed (without affecting their amplitudes in any way) to turn a sine wave at the input into a triangle wave at the output.

What I have described here is, of course, the basis of the frequency-modulation synthesis (or FM) used in Yamaha's DX series of synthesizers, and it is very similar to the 'Phase Distortion' (or PD) synthesis used in the Casio CZ series of keyboards. But instead of modulating an oscillator, as we did when investigating FM synthesis earlier in this series (see *SOS* April and May 2000 or www.soundonsound.com/sos/apr00/articles/synthsecrets.htm and www.soundonsound.com/sos/may00/articles/synth.htm respectively), we are now frequency-modulating *any* sound.

It's possible to build a mathematical model of the 'clock distortion' FM synthesis implied by Figures 9, 10 and 11 using a sine-wave oscillator to modulate the frequency of the clock (see Figure 12, left). There's nothing special about sine-wave

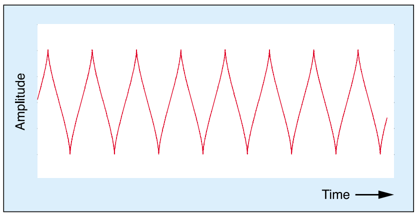


Figure 13: Modulating the output clock to shape a sine wave into a triangle wave.

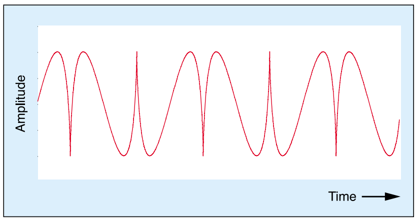


Figure 14: Obtaining a more complex wave by altering the LFO frequency in Figure 12.

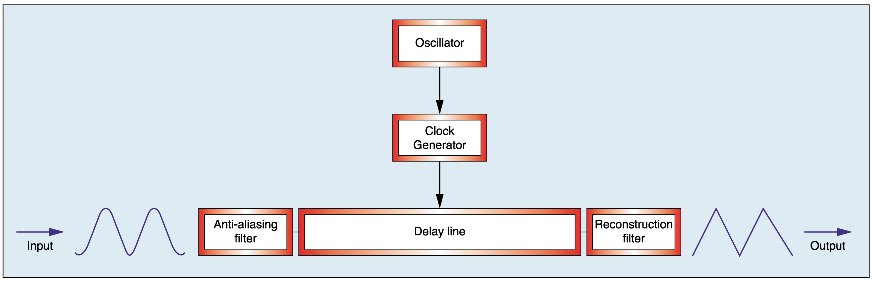


Figure 12: Modulating the clock.

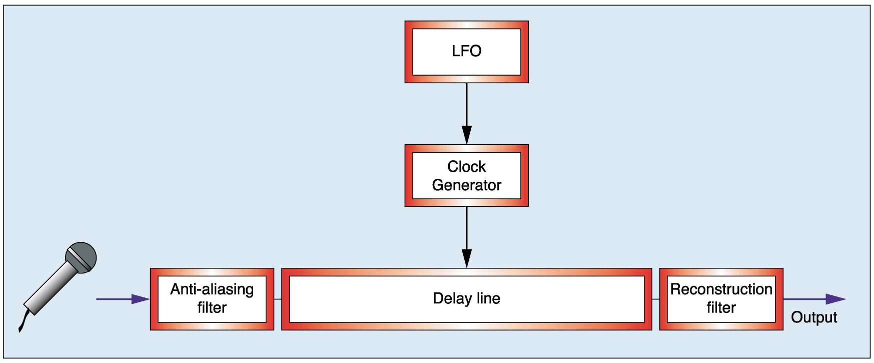


Figure 15: Simple pitch modulation of any audio signal.

modulation in this context — I could use any waveform — it's just that it's simple to implement a sine wave in a model of this nature. Using this, you can generate vibrato when the modulation oscillator is running in the LFO range, and many recognisable 'DX' and 'CZ' waveforms when it runs at audio frequencies. For example, Figure 13 (on previous page) shows the superb precision obtainable when using the modulator to 'waveshape' a sine wave into a triangle wave. Meanwhile, Figure 14 (on previous page) reveals that we can obtain a more complex-looking and harmonically interesting wave by modulating the clock at a different frequency. What happens when you use a delay line to 'FM' a complex signal such as your guitar playing or singing is, of course, another thing!

## Synthesizing The Leslie

Audio-frequency FM and PD synthesis are fascinating topics, but they are not the purpose of this month's Synth Secrets, so we have to leave them behind, return to the Leslie, and now ask what its modulation depth and frequency might be. Surprisingly, the modulation depth created by the doppler effect in a Leslie speaker is quite small — around ±1 percent, which would be no problem for the mechanism in Figure 12.

More problematic is the slowest rate at which the modulation occurs. For a Leslie rotor, this can be slower than 1Hz. This means that the modulation *depth* drops to a fraction of a percent, but the slow modulation frequency means that the delay line has to increase the audio frequency for half a second or more, and then reduce its frequency for half a second or more. This introduces some technical difficulties, often resulting in reduced signal integrity. Nevertheless, the principles of our analysis are correct, so I can draw a mechanism for imitating the doppler effect for any audio signal, as shown in Figure 15 (above left). As you can see, the audio is passed through the delay line and its associated filters, with the clock modulated at a low frequency, as discussed. The result is a signal that undergoes pitch modulation, no matter what ▶

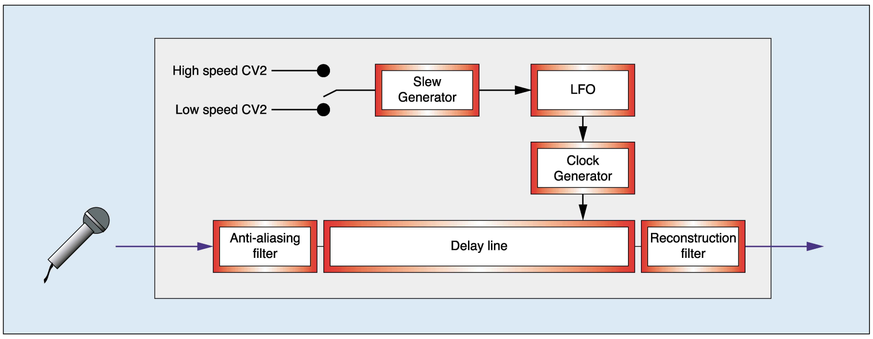


Figure 16: Providing two modulation speeds.

the nature of the input.

The depth and speed of the pitch modulation in Figure 15 are controlled solely by the LFO, and we can affect this by applying control voltages to that module's CV inputs. This leads to a number of interesting effects, one of which is the ability to use two CVs to imitate the Leslie's two rotation speeds, the 'tremolo' and 'chorale' mentioned last month. What's more, we can even control the rate of transition between these speeds by adding a slew generator that smoothes the transitions between the 'fast' and 'slow' CVs, thus emulating the acceleration and deceleration you hear when changing the speed of the physical rotors in the Leslie itself. (See Figure 16, above).

To make this model accurate, we must split the audio signal into two bands — a treble band above 800Hz and a bass band below 800Hz — just as in a real, dual-rotor Leslie speaker. The easiest way to do this is to split the audio into two signal paths and apply appropriate band-splitting EQs to each. We can then duplicate the modules in Figure 16, defining independent 'rotation' speeds and transition speeds within each path, as shown in Figure 17, below (which, for pertinence, I have drawn with a keyboard rather than a microphone as the signal source).

Now all we need to do is add the amplitude and tonal modulations discussed last month (see Figure 18, on next page) with each 90 degrees out of phase with respect to the LFO 'rotation' rate. I have added small delay lines in each of the control signal paths to generate this delay, but it is far from a complete description because, as the rotation rate changes, the lengths of these delays also need to change. This can be achieved by adjusting the clocks driving the secondary delay lines, but I suspect that you'll forgive me if I don't plumb the details of this.

Anyway, with all the delay lines, filters, amplifiers, LFOs, EQs, CVs and Slew Generators in place, we now have the glorious, analogue... argghh!! Figure 18 shows just *one* direct signal path for each rotor, without any of the reflections that occur within or outside the Leslie cabinet. To re-use last month's analogy, we have two roundabouts but no office blocks. Fortunately, a BBD is an appropriate device for creating simple reverberant effects so, in theory, the addition of another couple of delay lines (the fifth and sixth) might help to overcome this. But given the difficulties in getting this far, and the complexities I've just sidestepped regarding the phase relationships of the various modulations, I imagine that it's becoming clear why no analogue emulation of the rotary speaker cabinet was ever fully successful. To be fair, there was one — the Dynacord CLS222 — that was pretty damn good, and the effect on the Korg BX3 organ was useable if you were prepared to open the instrument up and tweak the internal trimmers.

## The Digital Leslie

For most people, the dream of a light, portable, inexpensive and

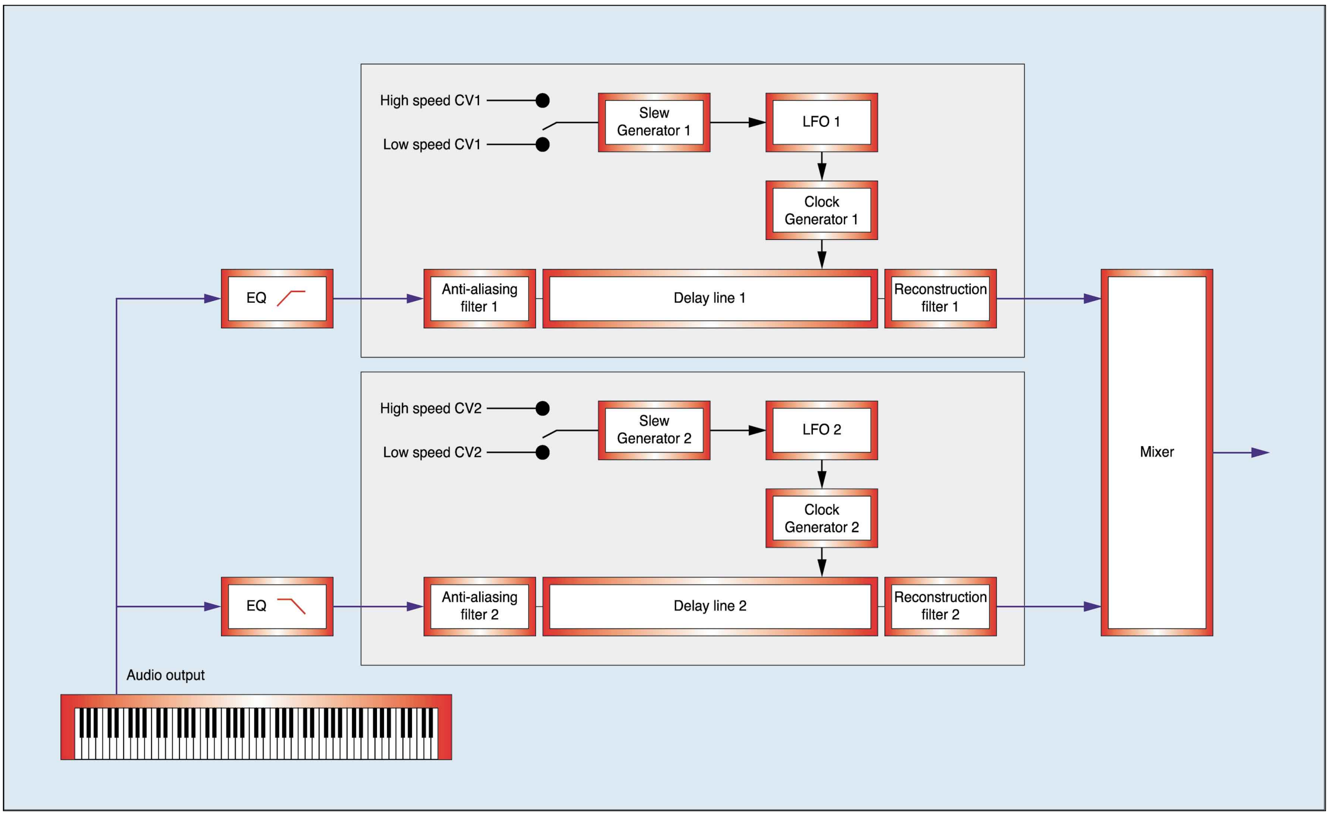


Figure 17: Modulating the upper and lower frequencies independently.

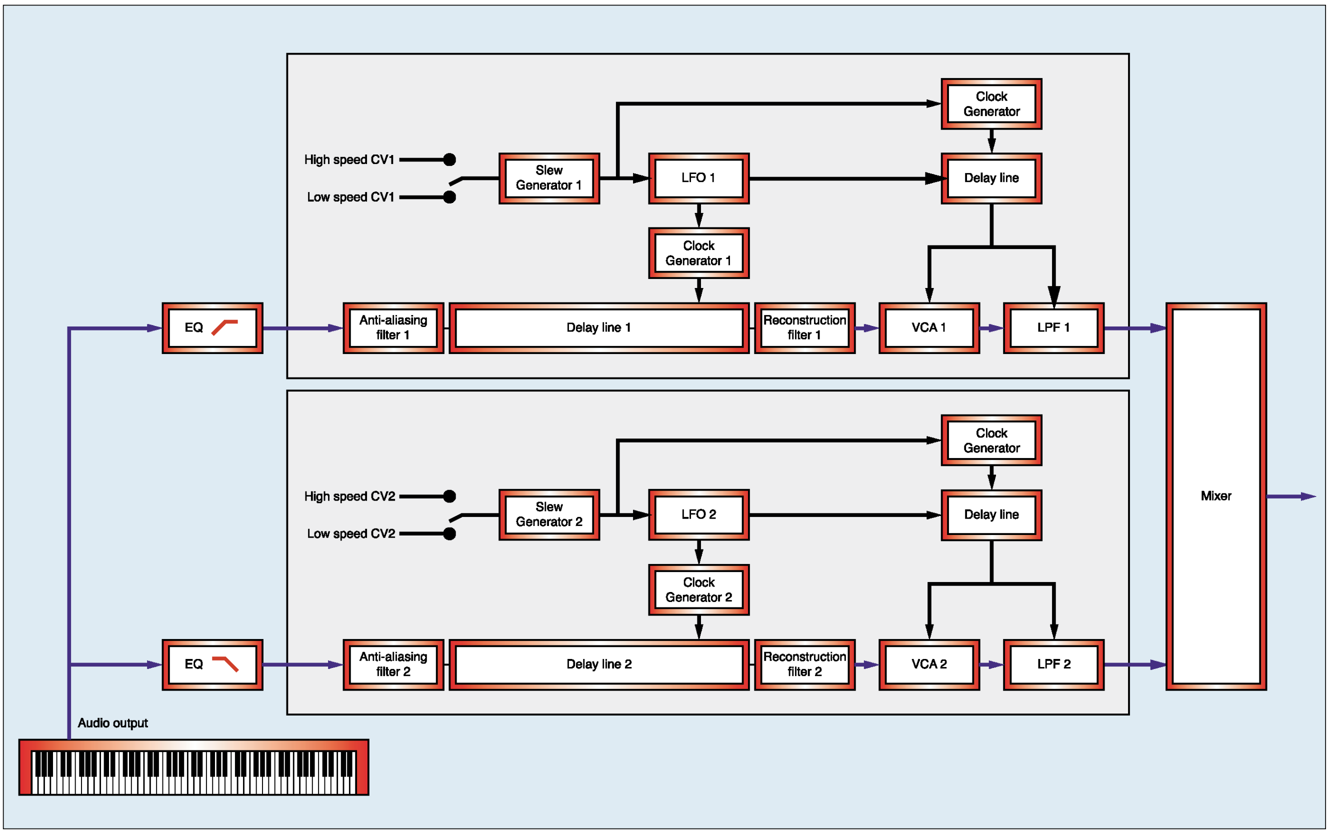


Figure 18: Attempting to create a dual-channel Leslie effect using delay lines.

authentic-sounding Leslie effect became a reality only with the advent of digital electronics, and the development of algorithms capable of modelling all the above factors successfully. These algorithms can calculate thousands of signal paths, each exhibiting different pitch shifts, different phases and different amplitudes. Sure, it takes a lot of processing power to implement them but, nowadays, that's not a problem.

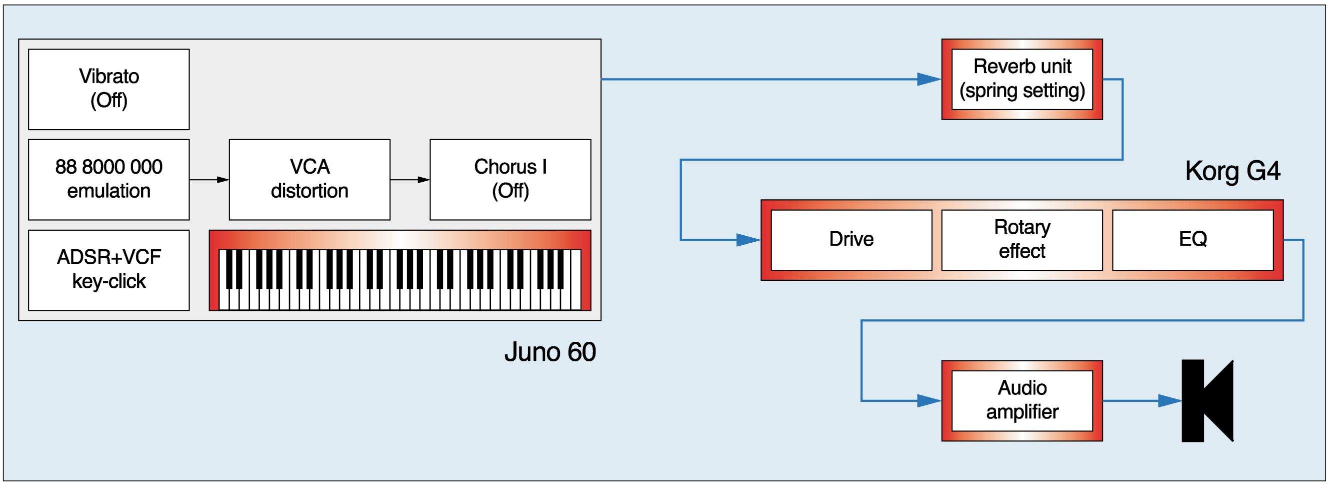
Of these, my favourite remains the Korg ToneWorks G4, a combined 'valve overdrive and rotary speaker' emulator. If you hook one of these up to a Juno 60 or the Kawai K3 I discussed a few months ago, the results are magic. The G4's overdrive is more realistic than the distortion imparted by the Juno's VCA, the rotary effect is remarkably authentic, and its speaker simulation gives it, in my opinion, just the right amount of dull woodiness. Connecting everything together, we obtain Figure 19 (above).

Of course, the algorithm in the G4 is synthesizable using analogue electronics, and with a wall of filters, clocks, modulation oscillators, delay lines and amplifiers, you could create a convincing electronic recreation of the rotary speaker effect. You would be mad to try, but you *could* do it.

## Epilogue

We have achieved a huge amount this month, learning how closely linked the seemingly disparate technologies of S&H, delay lines, phase-distortion synthesis, and digital converters prove to be. Moreover, armed with our new understanding of BBD delay lines, we could continue to develop our analogue 'Leslie' effect. Alternatively, we could extend some of this month's ideas to create all manner of effects, such as echo, flanging, chorus and ensemble. And that's what we're going to look at next month.

Figure 19: Using a digital Leslie emulator.



# Synth Secrets

Gordon Reid

## From Analogue To Digital Effects

**When synthesizing sounds, the effects you place *after* your synth's output are often as important as the synth itself (just think of last month's Leslie). As we near the end of Synth Secrets, we consider how a digital effects processor works.**

Last month, as part of the final push to synthesize the effects of the Leslie rotary speaker, I introduced the bucket-brigade device (or BBD) delay line and showed how we could attempt to use this to assist the simulation. I then showed that, while possible, this was not practical. In fact, because of space considerations, I omitted a number of secondary factors that make analogue recreations of the Leslie less than satisfactory. For example, the spatial amplitude response of the horn assembly is not smooth, so the volume of the sound 'wobbles' as lobes of loudness and quietness rotate past your ears. What's more, this response is frequency-dependent, meaning that there are independent amplitude modulations occurring for each frequency in the signal. Then there's the bass rotor... Due to its limited size, this is not effective at frequency-modulating low-frequency signal components so, unlike the horn, it is more a source of amplitude modulation ('tremolo') than frequency modulation.

All in all, it's little wonder that I gave up on my quest for the 'Analogue Leslie' and sent you away to find a Korg G4 or some other low-cost digital Leslie simulator. Admittedly, there are many simpler effects for which the perceived 'warmth' of analogue electronics is a bonus. For example, I don't think that anybody has improved upon the Electro-harmonix Deluxe Electric Mistress or MXR Flanger/Doubler, and even low-cost stomp boxes such as the Small Stone phaser and Big Muff occupy a unique place... often emulated, but only equalled if copied almost component for component. Nevertheless, for emulations of rotary speakers and the creation of new and esoteric effects, digital electronics is king. So, given that we're now nearing the end of our journey through the world of synthesis — from oscillators and filters at one end to the effects at the other — I think that it's time to introduce the fundamental electronic concepts that make digital audio possible.

### Analogue To Digital

We'll start by returning to the BBD that I explained last month, and which I've recreated in Figure 1 (below). This shows a signal entering the delay line through an anti-aliasing filter on the left of the diagram, passing through each stage in turn, and then exiting through the reconstruction filter on the right. The length of delay is determined by just two factors; the number of BBD stages and the speed of the clock driving the switches shown in red and green.

As we'll see, the elements that make up the digital equivalent of the analogue delay line are very similar in function. The principles at work in the BBD — of storing a slice of the incoming signal, holding it, and passing it down a line of identical, signal-storing components at a rate determined by a clock — are the same, although of course the means of storing the incoming signal is different. Instead of the signal being stored as a voltage in a capacitor at each stage of the BBD, it's a digital representation of the signal that is stored and processed, a binary number consisting of a string of ones and zeroes.

In order for this to be possible, the input signal has to pass through an analogue-to-digital converter (ADC) as it goes into the effect units. Most of us are now familiar with the concept of how one of these works; the instantaneous signal voltage is measured at intervals determined by the sampling rate (every 1/44100th of a second for CD-quality audio), then that voltage measurement is converted into a binary number composed of bits (the

▶

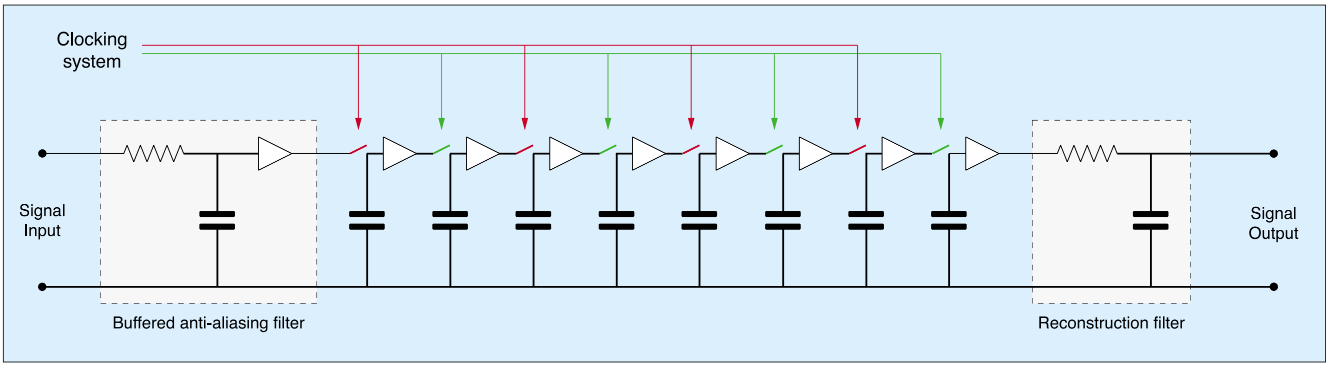


Figure 1: A simple eight-stage bucket-brigade device (or BBD) delay line.

number of digits in the binary number). The greater the number of bits, the higher the resolution of the signal measurement, and all other things being equal, this will enable you to represent the analogue voltage with smaller errors. CD-quality audio is 16-bit, so the voltage measurements are stored as strings of 16 digits, all of which are either a zero or a one.

I'm now going to explain what happens to a single one of those bits in a digital delay line — just one of the digits that makes up the binary measurement of the original analogue signal in one sample. In short, it passes through the digital equivalent of the BBD, which is known as a 'shift register', and which is itself constructed from devices called logic gates. However, to understand how a shift register works, we must first take a diversion into the fundamental nature of binary numbers and logic. Hold onto your hat... this is going to take us into a fascinating realm of technology not yet plumbed by Synth Secrets. If you'd rather not have your hat disturbed by the finer points of how logic gates and shift registers work, you can skip ahead to the heading 'An Audio Delay Line', on page 130.

## Combinational Logic

Firstly, it's all very well to say that the analogue signal is converted by the ADC into a string of ones and zeroes, but what does that actually mean? After all, how does a '1' or a '0' pass through an electronic circuit? Well, perhaps strangely, given that the input signal started off as a voltage, the zeros and ones that make up a 16-bit sample are also voltages, albeit used differently. In this case, any voltage above a predetermined level 'v1' is said to be a '1', while a voltage below another predetermined level 'v0' is said to be a '0' (see Figure 2, below).

Actually, this is an idealised view, and to make such a system work, the change from '0' to '1' generally occurs when the voltage passes from below v1 to above v1, and a transition from '1' to '0' occurs when the voltage drops from above v0 to below v0. So the 16-bit numbers that are output at each clock step of the ADC are represented by 16 voltages passing through the effects unit, each of which is understood by the circuits through which it passes as a '1' or a '0'. And what are these circuits? As I said earlier, they're logic gates, whose operation is predicated on their ability to determine whether their inputs are in one state or another: zero or one.

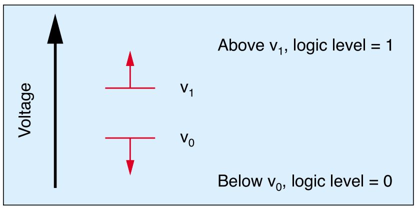


Figure 2: The nature of a digital system.

One of the simplest forms of logic gate is a device called the two-input AND gate, which I've represented in Figure 3 (left), and whose operation I have described in the 'truth table' shown below. If you look at this, you'll see that both the level presented to input 'X' *and* the level presented to input 'Y' must be '1' for the output of the device to be a '1'. Any other combination results in an output of '0'. The circuitry required to do this is remarkably simple, but explaining it would take us off into transistor electronics, which is not where I want to go, so we'll say no more about it here.

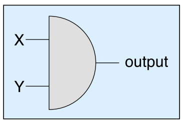


Figure 3: An AND gate. Note that the gate symbols I have used in this article are a mixture of traditional British and American standards. You may encounter others elsewhere.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 0 | 0 |
| | 1 | 0 | 1 |

Table 1: The truth table for a two-input AND gate.

There are numerous other types of logic gate. For example, there's the OR gate, which, in its two-input form, produces a '1' when either of X *or* Y are '1' (see Figure 4, below).

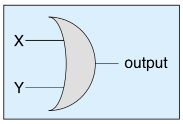


Figure 4: The OR gate.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 0 | 1 |
| | 1 | 1 | 1 |

Table 2: The truth table for a two-input OR gate.

Next comes the NOT gate, or 'inverter' which has a single input, and produces a '0' when the input is '1', and *vice versa* (see Figure 5 below).

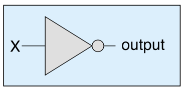


Figure 5: The NOT gate.

| | | |
|---|---|---|
| INPUT | 0 | 1 |
| | 1 | 0 |

Table 3: The truth table for a two-input NOT gate.

If you consider the action of an AND followed by a NOT, it follows that there is another device, called a NAND gate, that responds to the logic 'and not', meaning that a '1' is output when [input X *and* input Y] is *not* '1' (see Figure 6, above).

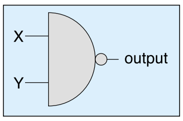


Figure 6: The NAND gate.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 1 | 1 |
| | 1 | 1 | 0 |

Table 4: The truth table for a two-input NAND gate.

Likewise, there is a NOR gate which acts as an OR followed by a NOT (see Figure 7, below).

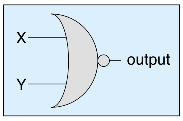


Figure 7: The NOR gate.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 1 | 0 |
| | 1 | 0 | 0 |

Table 5: The truth table for a two-input NOR gate.

Next, there's the XOR ('exclusive OR') gate, which produces a '1' when *either* X *or* Y are '1', but not when both are '1' (see Figure 8, below).

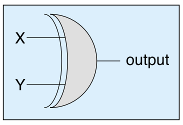


Figure 8: The XOR gate.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 0 | 1 |
| | 1 | 1 | 0 |

Table 6: The truth table for a two-input XOR gate.

Finally, there's the XNOR, which is the inverse of the XOR. This outputs '1's when the inputs are both '0' or both '1', but not otherwise (see Figure 9, below).

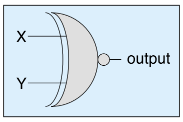


Figure 9: The XNOR gate.

| | | INPUT Y | |
|---|---|---|---|
| | | 0 | 1 |
| INPUT X | 0 | 1 | 0 |
| | 1 | 0 | 1 |

Table 7: The truth table for a two-input XNOR gate.

Once you have these seven devices at your disposal, you can design anything from a simple logic switch to the most complex computer. You don't even need all seven types because, with just AND, NOT and OR, you can derive all the others. Although this may not be the most efficient way to obtain a given result, it means that many complex problems can be reduced to a simpler form.

## Gates With Memory

What we have discussed so far is called 'combinational logic' because the inputs at any moment determine the values of the outputs. In other words, individual gates and many of the systems developed from them have no 'memory'.

Fortunately, we can build a 'sequential circuit' — one whose output is not only dependent upon the current inputs but also on *past* inputs — by connecting the outputs of two NAND gates to each other's inputs. This forms a feedback loop that, for a given set of inputs, snaps into one state or another, and then holds this state after the inputs are removed. The circuit thus formed is known as a flip-flop.

How does it work? Well, consider two NAND gates connected as shown in Figure 10 (below). This configuration is called an RS flip-flop, where the letters 'R' and 'S' stand for Reset and Set. (This circuit is also called an SR flip-flop, and we can draw it in different ways, but the logic is always the same.)

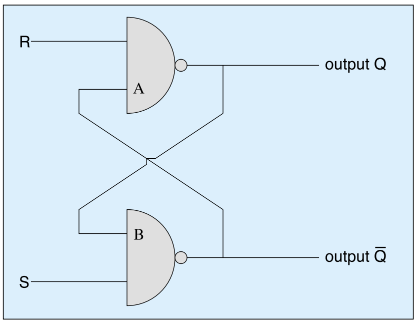


Figure 10: The RS Flip-flop.

Imagine that the R and S inputs in Figure 10 are both '1'. If we then apply a '0' to the R input the output 'Q' will be forced to be a '1'. Conversely, if we apply a '0' to the S input, the output will be forced to be a '0'. Does this sound like gobbledygook? If so, let's work out what's happening. We'll start by considering the two possible initial states for the system when both R and S are '1'.

- Firstly, then, if the input 'A' on the upper NAND gate is a '0', the output Q must be a '1'. You can check this by looking back as the NAND truth table (in Table 4 on page 126) if you like. Sure enough, if one of the inputs to the gate is a '0', then irrespective of what the other input is, the output has to be a '1'. This means that the fed-back input to 'B' must also be a '1'. Consequently, both the lower inputs 'S' and 'B' are '1', so the output Q̄ (which means the opposite of Q), reading from Table 4 again, is a '0'. Q̄ is, of course, the '0' fed back to the 'A' input on the upper gate, so the logic of the system is self-consistent.
- The other possibility is that we could have A= '1' alongside the R= '1' input, in which case Q must be '0', and Q̄ must be '1'. Either way, the logic works, and the system is stable.

Now, if we apply a '0' pulse at the R input, we create a situation where it doesn't matter whether A is '0' or '1'... The output Q must become a '1', and we say that it has been Reset. With S= '1' and B = '1', Q̄ must be a '0', so we now know that 'A' must be a '0'. The logic is consistent and unambiguous, and the system has a single, stable output state.

The interesting thing about this logic system is that the Reset pulse can be very brief; if the input at R returns from '0' to '1' again, Q remains '1' and Q̄ remains a '0'.

Once the system has been Reset in this way, the one thing that changes the output state is a 'Set' pulse of '0' applied to the S input. With S= '0' and B= '1', the output Q̄ becomes a '1', and Q becomes '0'. And, as with the Reset pulse, the Set pulse can be as brief as you like. Once flipped into this state, the flip-flop's outputs remain constant until Reset again.

It should now be clear how the RS flip-flop came by its name. As long as R and S are not pulsed to '0' at the same time (which makes the outputs indeterminate) the device flips and flops between two stable states.

If we're to make the RS flip-flop useful, the next thing we need to do is to ensure that the device only flips (or flops) at specific times. We do this by adding a clock input and another couple of NAND gates, as shown in Figure 11 (below). This device is called a Gated RS flip-flop, and it should be obvious from the NAND truth table that the levels at the inputs X and Y can only affect R and S (and, therefore, the outputs Q and Q̄) if the clock pulse — which is sometimes referred to as 'Enable' — is a '1'. If, instead, the clock is a '0', the output from the gates with the X and Y inputs will always be '1', irrespective of what X and Y are.

Now, without stepping through all the possible logic levels as I did for the basic RS flip-flop, I can write the truth table for the Gated RS flip-flop, as shown in Table 8 (below). This is a little more complex than before, because we have to consider not only what happens at R and S, but how this is affected by the inputs X and Y. On the table, Qn simply means 'whatever Q is after an arbitrary number of clock pulses'. The table then shows that, with X='0' and Y= '0', the output is the same as it was after the clock's previous pulse, whether that was a '0' or a '1'. It doesn't matter whether the output Q is a '0' or a '1'; provided that X and Y are both '0', it remains unchanged for all subsequent clock pulses. However, we can load another 'bit' of information at any future time by applying a '1' at X or Y, as appropriate. *Unlike combinational logic systems, the Gated RS flip-flop possesses a programmable memory!*

| X | Y | R | S | Qn |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | Q at previous clock pulse |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | ? (indeterminate) |

Table 8: The Gated RS flip-flop truth table.

## The D & JK Flip-Flops

By this point, I imagine that you have either given up trying to understand what's happening, or you've jumped out of the bath yelling 'by Jove, I think I've got it!'. Nevertheless, we must discuss a couple more steps before we can talk about digital audio and delay lines.

Firstly, we have to get rid of the indeterminate state that exists when both the X and Y inputs in the Gated RS flip-flop truth table are '1'. It's simple to correct this: we redesign the device so that it has just one input, called 'D' (for Data) as shown in Figure 12 (on the next page).

Unfortunately, while the inverter on the inputs ensures that the

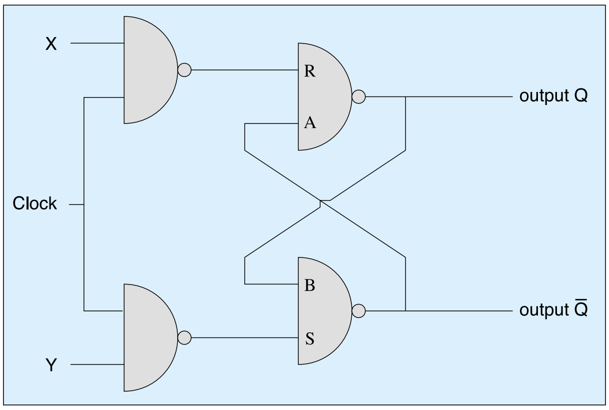


Figure 11: The Gated RS flip-flop.

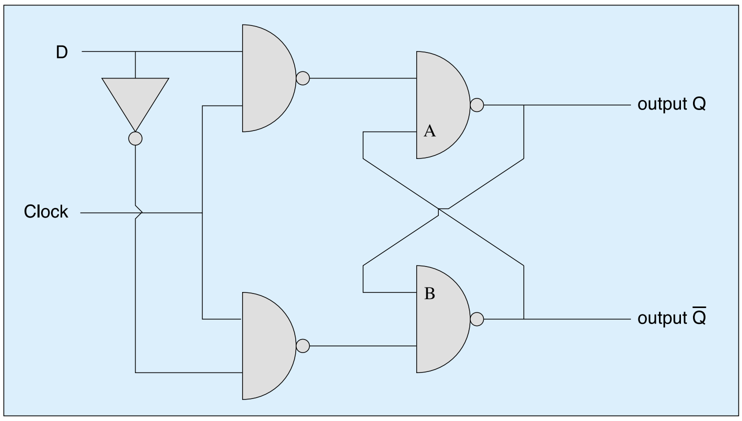


Figure 12: The D flip-flop.

▶ indeterminate '1,1' state can never occur, it also means that the '0,0' state that holds the bit of information until reprogrammed is also impossible. That's not to say that the D flip-flop has no applications... far from it, because, whenever the clock is '1', the output 'Q' takes the value at 'D' and holds it until the next clock pulse. As you will see, there are many uses for this, but if we want a programmable memory *without* the indeterminate state, we must take a final step in our journey into the world of digital logic, with the introduction of the JK flip-flop (see Figure 13, below).

This device introduces the concept of three-input NAND Gates, with the truth table shown in Table 9 below. As you can see, this is just an extension of the table shown in Figure 4, and we could extend this further for any number of inputs we desired.

| X | Y | Z | Q |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Table 9: The truth table for the three-input NAND gate.

The input names 'J' and 'K' don't have any significance, but the meanings of the 'R' and 'S' inputs remain unchanged. To Reset the single bit of memory in the device to '1', you need only apply a brief '0' pulse to the R input. To Set the memory to '0', apply a brief '0' pulse to S. With R and S both set to '1', the truth table in Table 10 applies. Again, I don't propose to step through every possible logic state to prove this, but if you get a piece of A4 and a very sharp pencil, you should be able to derive it for yourself.

| J | K | Qn |
|---|---|---|
| 0 | 0 | Q at previous clock step |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | $\overline{Q}$ at previous clock step |

Table 10: The truth table for the JK flip-flop.

The great thing about the JK is that it not only removes the ambiguous state from the Gated RS flip-flop's truth table, it converts it into something useful: a toggle from the existing output value to the next one. Finally, let's simplify Figures 12 and 13, by adopting logic symbols for the D and JK flip-flops, as shown in Figures 14 and 15 (right), before a quick recap.

## An Audio Delay Line

If you skipped most of this article to get to this point, you've missed some pretty heady stuff. It's your loss, but we can summarise what we've learned as follows:

- The D flip-flop is a device that takes an input, holds it, and passes it to its output.
- The JK flip-flop is a device that allows you to determine the state of its output 'Q', and either store this indefinitely, redetermine it, or toggle it, as desired.

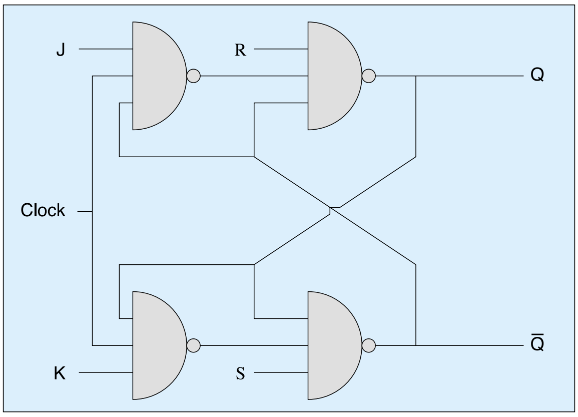


Figure 13: The JK flip-flop.

With these two devices, we can now build a one-bit 'shift register' that allows us to determine the value of a single bit at the start of the line, and pass it through numerous elements, as shown in Figure 16 (on the next page).

Of course, in a 16-bit digital audio effects processor (which, of course, is where we came in), a delay line will require 16 such registers arranged in parallel, fed by the ADC on the input as discussed at the start of this article. At the far end of the delay line, a digital-to-analogue converter (DAC) will convert the data back into an analogue audio signal.

Actually, since the input value of each 'bit' will be determined by the ADC in our effects unit, we don't need to use a JK flip-flop as the first element in the delay line; we can use the simpler D flip-flops throughout (see Figure 17).

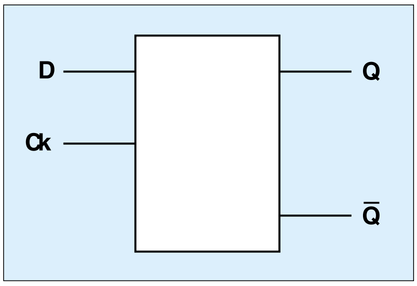


Figure 14: The logic symbol for the D flip-flop.

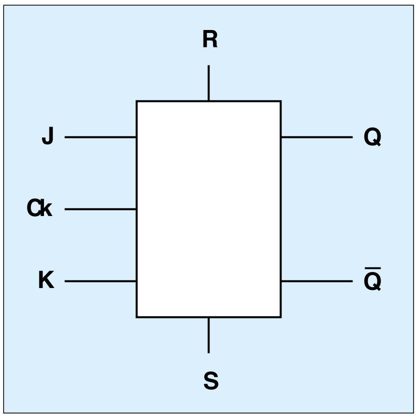


Figure 15: The logic symbol for the JK flip-flop.

To demonstrate this, let's consider an example in which the ADC provides a '1' to one of the registers in Figure 17 on the first clock pulse, followed by three '0's. If the rest state of the D flip-flops is '0', the data will pass down the line as shown in Table 11 (on next page).

Naturally, we can expand this concept to include 16-bit information, or 24-bit, or even 64-bit, simply by adding the requisite number of parallel registers, so in principle it is straightforward to pass high-resolution digital audio from the ADC to the DAC.

|  | ADC | D1 | D2 | D3 | D4 |
|---|---|---|---|---|---|
| Initial state | 0 | 0 | 0 | 0 | 0 |
| Clock 1 | 1 | 1 | 0 | 0 | 0 |
| Clock 2 | 0 | 0 | 1 | 0 | 0 |
| Clock 3 | 0 | 0 | 0 | 1 | 0 |
| Clock 4 | 0 | 0 | 0 | 0 | 1 |
| Clock 5 | 0 | 0 | 0 | 0 | 0 |

Table 11: Passing data down a shift register.

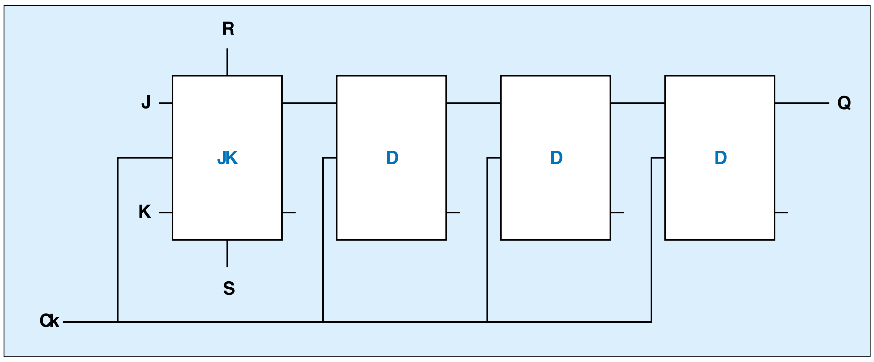


Figure 16: A one-bit, four-element, digital delay line.

So there we have it... Figure 17 is recognisably equivalent to the BBD shown in Figure 1. As before, the signal enters the delay line through an anti-aliasing filter on the left of the diagram (although, in this case, it's part of the ADC) passes through each stage in turn, and then exits through a reconstruction filter on the right (which, in Figure 17, forms part of the DAC). And, as with the BBD, the length of the delay is determined by just two factors; the number of delay stages, and the speed of the clock that's driving the signal through the devices.

## Epilogue

It is, perhaps, harder to grasp the principles of flip-flops than it is to understand a handful of resistors, capacitors, and analogue switches, but once you have done so, the possibilities become enormous. Depending upon what additional gates and connections are added to the shift register, it is equally at home moving data from right to left as it is left to right, and will accept data in parallel and output it in parallel. Furthermore, if we were to replace all the D flip-flops in the register with JK devices, we could 'program' the value held by each one independently and, at some future point, read back any combination of these values independently. This, of course, is Random Access Memory, or RAM. Furthermore, if we were fix the levels of the inputs to the JKs, the output values would be predetermined, and we have a Read-only Memory, or ROM. Indeed, the concepts explained this month underlie every aspect of digital technology. From the simplest electronic switches to the 16-, 32- or even 64-bit keyboards, effects units, mixers and computers that we encounter in every walk of our musical lives... they're *all* based on these ideas.

Let's face it, that's not bad for a bunch of zeros and ones. SOS

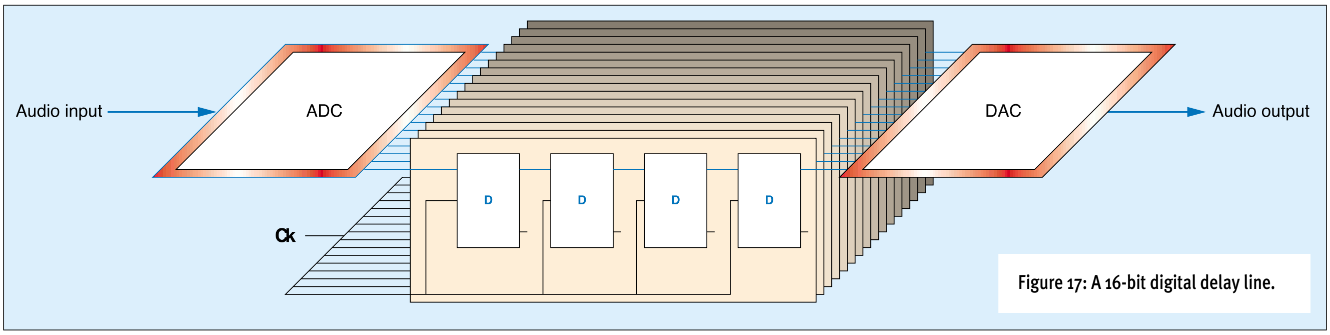


Figure 17: A 16-bit digital delay line.

# Synth Secrets

Gordon Reid

## Creative Synthesis With Delays

**Effects can play just as important a role in sound creation as the elements in a synth's signal path — provided you have access to their constituent parts. We take a closer look at effects synthesis with simple delays.**

For the past two months, I've been describing the nature of delay lines; what they are, and how they work. Having done so, I'm going to make good my promise to show how we can use these to create many of the effects used in today's music, starting with various forms of echo and an unusual reverberator.

You may be wondering what place this discussion has in a series on synthesis. Well, firstly, most synthesizers have some form of effects unit(s), and I'm not just talking about digital workstations with their zillions of insert and master effects. From the earliest days of synthesis, experimental modular instruments offered spring reverbs, and many of the revered monosynths from the 1970s — the ARP 2600, EMS VCS3, and umm... the Teisco S100P — did likewise. Admittedly, the practice lost favour for a while, but when bucket-brigade devices (or BBDs) became affordable, the effects that they made possible signalled the introduction of new breeds of synthesizer. Most obviously, ensembles or 'string machines' appeared in large, usually Italian, herds. Simultaneously, chorus units and flangers (which are also effects using delay lines) started to appear on instruments such as the ARP Quadra, the Korg Trident, and the Roland Jupiter 4. But I think that there's an even more important reason to discuss effects here.

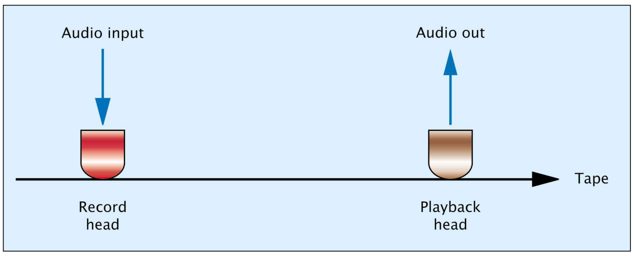


Figure 1: The simplest electronic delay represented as a tape delay.

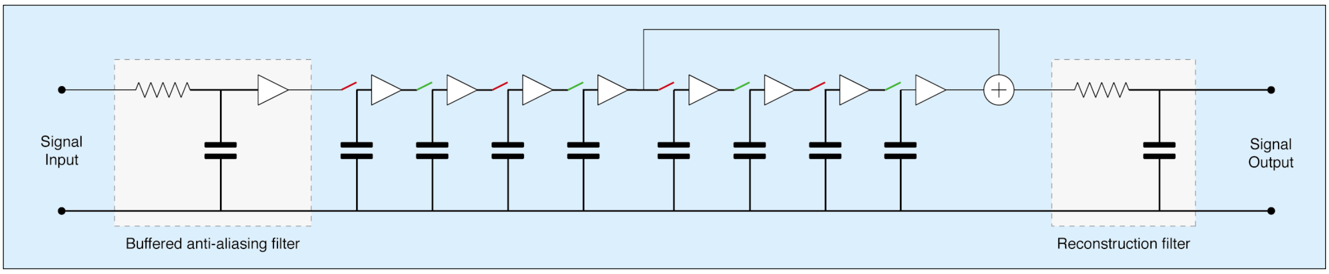
Just as modern synthesizers provide hundreds or even thousands of presets, and have been blamed by many for the demise of innovative sound programming, modern effects units have scores of presets, and can equally be blamed for the demise of innovative *effects* programming. Indeed, many modern effects units — from the humble stomp box to sophisticated studio systems — offer a fixed architecture, and turning the knobs just changes the values of the parameters within that architecture. Some recent products offer variable architecture, allowing you some freedom in how you order different effects blocks in your effects patches, but this is still not the same as having access to the basic building blocks of effects synthesis, and being able to build new, innovative effects structures. Given the huge transformations that effects can wreak on your synth sounds, if you're reading Synth Secrets for ideas on creative sound programming, surely you should extend the same philosophy to your effects?

### Analogue, Digital & Tape Delays

I'm going to start looking at what you can do with delays by returning to an analogy I made in passing two months ago. In its simplest incarnation, you can think of a delay line as a tape recorder with an infinitely long strip of tape passing across a record head and then across a replay head (see Figure 1, left). The length of the delay is determined by the distance between the heads, and the speed of the tape as it runs between them. As analogies go, this is a good one.

If this were the only type of delay that we could create using electronics, it would not be a very interesting piece of technology. But let's now imagine that we can 'tap' off the signal before it reaches the end of the line. This would enable us to create two delayed signals — one that occurs as the input signal passes the first tap, and another as it reaches the end of the line. The analogue circuit for this is shown in Figure 2 (below), in which the circle with the 'plus' sign inside is a mixer. I'm sure that you can

Figure 2: A two-tap delay line.



imagine how the digital equivalent would look; it would require a data tap in each register between the fourth and fifth 'D' flip-flops on the schematic I drew at the end of last month's instalment of this series. But since I would have to invent the 16-bit digital mixer to draw this for you, I think that I'll stick with analogue for the moment!

If I redraw Figure 2 in standard Synth Secrets fashion, I create Figure 3 (left), which shows the delay elements on either side of the tap as two separate delay lines. This is not as strange as it may seem... there are numerous products that arrange two BBDs in series to generate delays of the required length, with or without taps between them. By the way, you may have noticed that I've removed the anti-aliasing and reconstruction filters from Figure 3, but that's not because I've removed them from the circuit. I'd like you to assume that they are in place, because I'm going to omit them from all the following diagrams for the sake of simplicity.

Let's now consider what Figure 3 does to audio passing through it. If we present

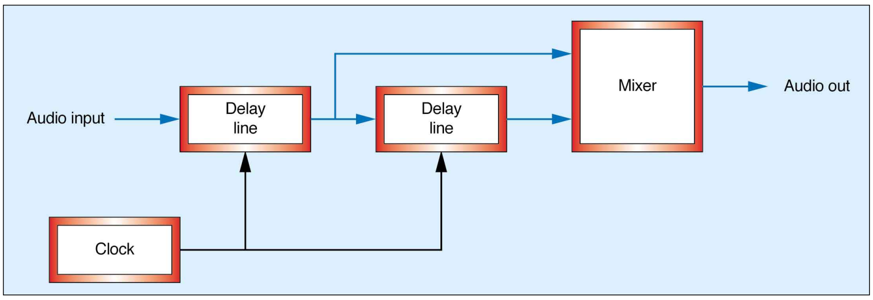


Figure 3: The two-tap delay line shown as a block diagram.

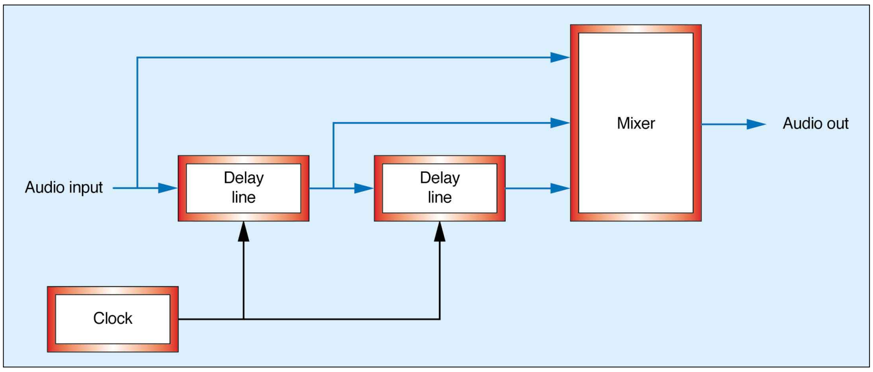


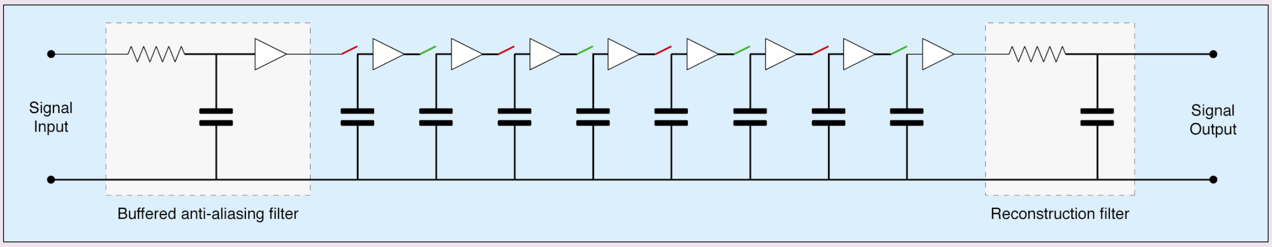
Figure 4: Adding the original signal to the delays.

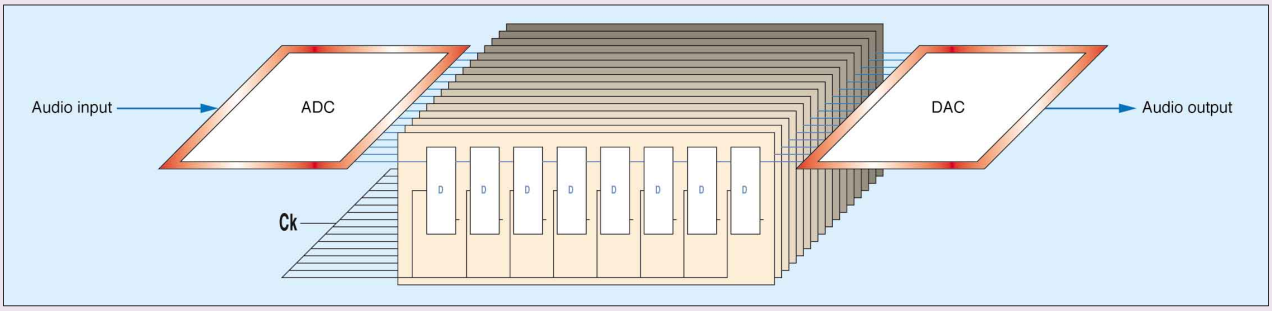
## Digital & Analogue — What's The Difference?

**Having spent much of last month's Synth Secrets explaining the operation of a digital delay, and then pointing out how conceptually similar this is to an analogue bucket-brigade device, it's important to point out that they often sound very different. Most of us know this, of course, and know too which side of the analogue versus digital debate we come down on. But this raises a question whose answer is often assumed, but rarely explained. It's this: "Given the conceptual similarities between analogue and digital delays, why is it that they often sound so different?".**

**The diagrams below show the aforementioned delays; an analogue BBD with its anti-aliasing and reconstruction filters, and a digital delay line with its associated A-D and D-A converter. As you can see, the two are equivalent, with a stream of samples moving from the input on the left to emerge unmodified on the far right. So, if the sample rate and number of stages are the same in each, why *do* they sound so different?**

**The answer lies in the degradations that occur as the signal passes down the line. If there is no failure in the digital delay line, the same data will arrive at the D-A converter as left the A-D converter, so the only changes that occur when comparing the audio input to the audio output (other than delay, of course) are those imposed by the limitations of the converters themselves. In contrast, each Sample & Hold stage in the analogue bucket-brigade device will be affected by the limitations of the capacitors and by electronic noise, so each stage will add or subtract a small voltage from each sample. These errors are cumulative, and although an amount of positive voltage noise added in one place might be cancelled out by a bit of negative voltage noise in another, every sample will be modified by the time it reaches the reconstruction filter. If the errors are random, the resulting signal will sound the same as the original with the addition of white noise, but more often than not, there will be some form of systematic error introduced. Whether you view these differences between the input and the output as a problem or a benefit, however, depends on the kind of sound you favour, and possibly the prevailing wind of current fashion!**





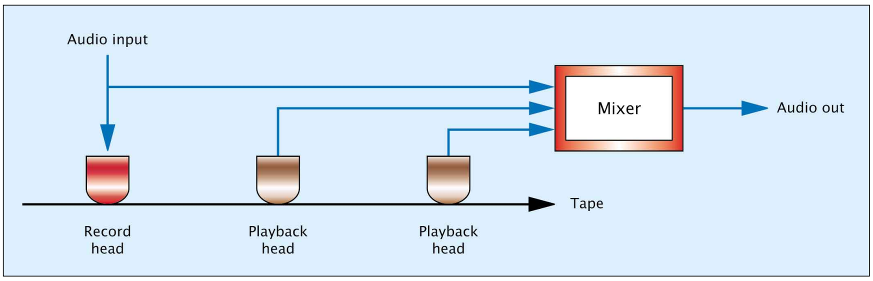


Figure 5: Figure 4 implemented as a tape-delay system.

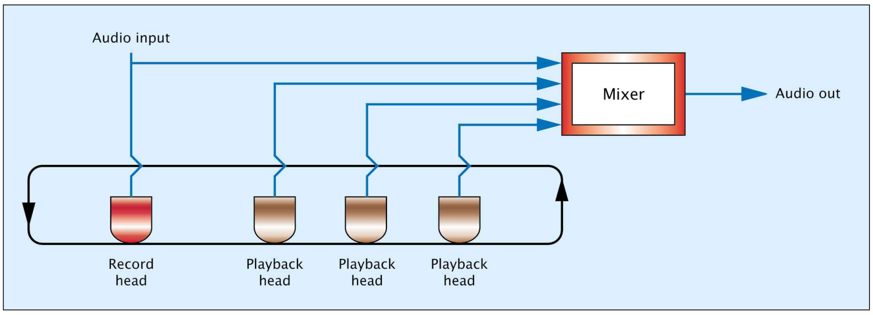


Figure 6: A tape-echo loop.

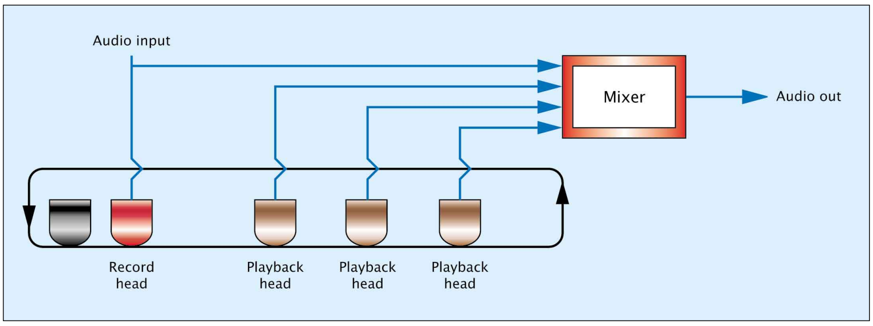


Figure 7: A usable tape delay system.

▶ a single impulse (a 'ping') to the input, we obtain 'pause... ping...pause...ping' at the output. Interesting though this is, it's not very musical, because the initial 'ping' — whether sung or played on an instrument — is lost, and if you played this on the beat, the first 'ping' that you would hear would trail the beat by the length of the first delay line. So we add another signal path that carries the original signal directly to the mixer, as shown in Figure 4 (on the previous page). Now we obtain 'ping...pause... ping...pause...ping', which is much more musical. The tape delay achieves the same effect by using two playback heads and directing part of the input signal to the output, as shown in Figure 5 (top). At this point, it's worth noting that there's nothing stopping you from making the lengths of the two delay lines — or the distances between each of the tape heads — unequal, so this system also allows you to produce syncopated delays. Now we're getting somewhere!

Developing this idea further, many tape echo units possess more than two playback heads, with three or four being the norm. However, they do not use infinitely long strips of tape, substituting instead a loop that passes endlessly over the record and playback heads, as shown in Figure 6. Unfortunately, this idea has a serious flaw. If you were to record a 'ping' on to the tape, it would be tapped off by the three playback heads, then reappear at them a short time later, and then again, and again, and again... *ad infinitum*. If further pings are added as time passes, you end up with sonic mush. Likewise, if you record a continuous signal such as a voice or a guitar, mush again ensues.

The answer to this problem is to add an erase head that removes some or all of the signal from the tape each time it passes (see Figure 7, left). If it erases the tape fully, we obtain just three delayed pings. If it erases the tape partially, we hear groups of three pings, with each successive group diminishing in loudness until they disappear into the noise floor. As you would expect, this sounds rather pleasing, so this architecture has become the basis of all professional tape-delay systems.

You can achieve the same result using delay lines, although the block diagram for this (see Figure 8, below) is already starting to look a little complex. The key element is the amplifier in the feedback loop, which is often labelled 'Regeneration' on basic echo units. With a gain that ranges from 0 percent (which is analogous to total erasure) through to 100 percent (which would be the situation if the erase head were removed) you can create the same range of effects as you would obtain from a tape-echo system. In practice, the degradation in BBDs will soon turn a re-re-re-repeated 'ping' into a sonic splodge, as will a tape that is being partially erased each time it loops, so there is a practical limit to the number of repeats you can obtain unless you use digital delays — which is, of course, what most of us now do.

## A Modular Delay Unit

We could of course recreate the effect of Figure 8 in a modern effects unit quite quickly, but although that would be simple,

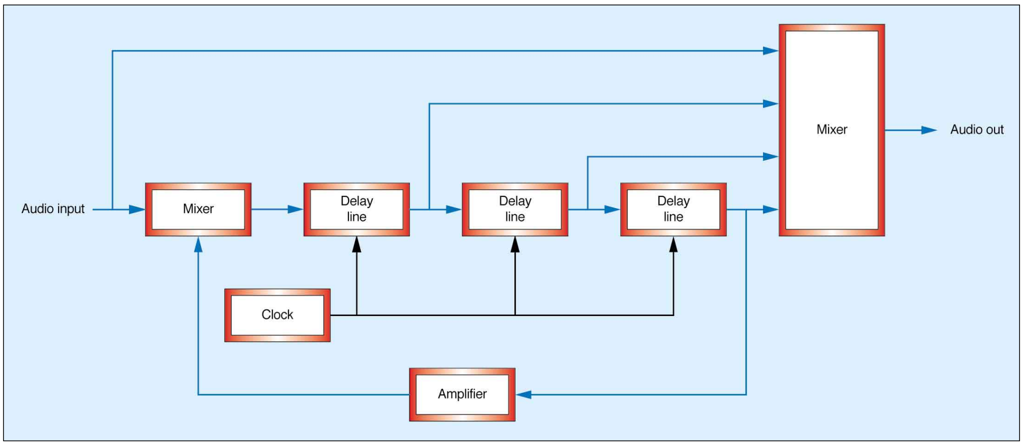


Figure 8: Representing the tape delay as a block diagram.

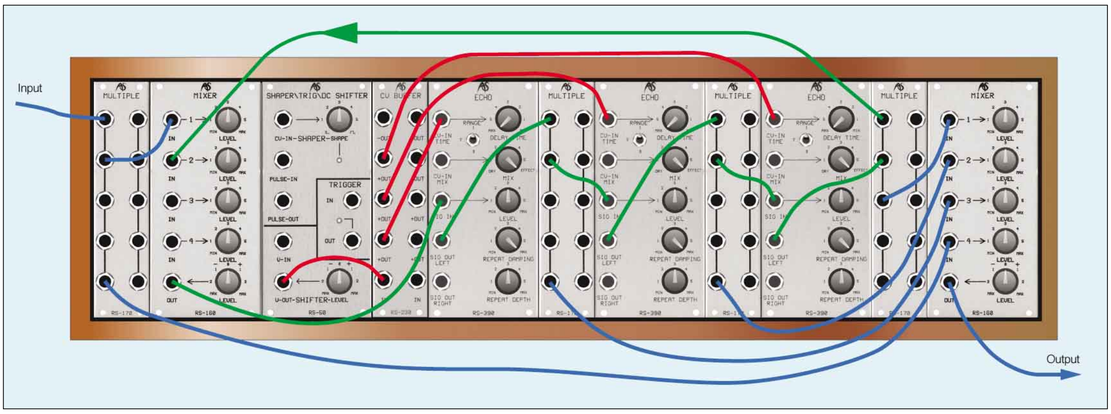
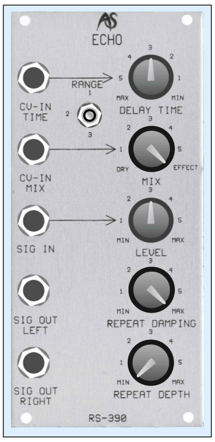


Figure 9: Creating Figure 8 using synthesizer modules.

some of the effects I'm planning to talk about require a level of control not available in all effects units. So let's consider instead how we might go about recreating Figure 8 inside a synth (see Figure 9, above). In my case, this means turning to my trusty Analogue Systems RS Integrator modular system, but of course there are plenty of other solutions, in both analogue modulars and modern digital or software-based synths, with which you can achieve the same degree of control.

However, the delay module *I've* chosen is the RS390, a digital delay line with dual (left/right) outputs, and user control over the relative loudness of the original and delayed signals. To construct the patch correctly, we need to select just one output from each of the three modules (I've chosen the left output), set the Mix control to 'Effect', the Repeat Depth control to its minimum and the Damping control to maximum, as shown in Figure 10 (see above). The first of these ensures that we obtain nothing but the effected signal from each module, and the others ensure that the internal regeneration of each RS390 is set to 'zero'. Each of the RS390 delay lines has an internal clock, so I have synchronised them by setting all the Delay Time knobs to the same position ('Max') and then controlling them using a single voltage derived from the CV generator/shifter in an RS50. OK, I admit that it's not the simplest of patches, but if you make all the connections — blue and green for audio, plus red for CVs — you'll find that it, or something akin to it on *your* synth of choice, recreates Figure 8 accurately.

Figure 10: Setting the RS390 to act as a simple delay line.



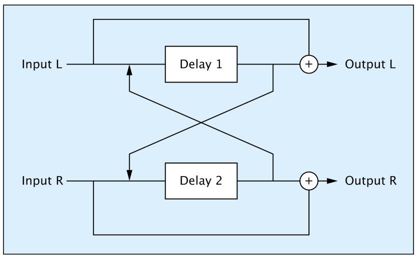


Figure 11: A 'ping-pong' delay.

Firstly, consider the audio, which I have divided into two parts: blue cables for I/O, and green cables to denote the path through the delays as well as the regeneration loop.

The input is distributed to two destinations. The upper cable takes the signal to a mixer (the second module from the left), while the lower takes it to the output mixer on the far right. The 'upper' signal passes from the mixer to the first echo module, the output from which is passed via a multiple to the second echo module *and* to the output mixer. Likewise, the output from the second echo module passes to both the third echo module *and* the output mixer. The output from the third echo module passes via the rightmost multiple to the input mixer on the left of the figure (thus creating the regeneration loop) *and* to the output mixer.

Secondly, consider the control voltages, which are denoted by the four red cables. The CV Shifter allows you to determine a control CV for the Delay Times of each of the echo units. I have directed the output from this to a CV buffer that then distributes the voltage to the CV-In Time inputs on each of the delays, thus allowing us to control all three units using a single control. Of course, there's nothing stopping us from creating offsets between each of the delays, simply by setting the Delay Time knobs and/or Ranges differently on each.

This 'modular effects unit' can create all manner of sophisticated delay effects, and can be modified further to generate an even wider range of sounds. For example...

Consider Figure 11 (above left), the standard schematic for a 'ping-pong' delay. This is a superb effect, and although it was esoteric back in the heyday of analogue synths, it has now been made commonplace by the proliferation of affordable digital stereo multi-effects units. Figure 12 (below) shows an RS390 patch to produce this effect. You'll notice that you need more mixers and multiples than Figure 11's representation suggests, but if you follow all the cables (blue for the left channel, green

▶

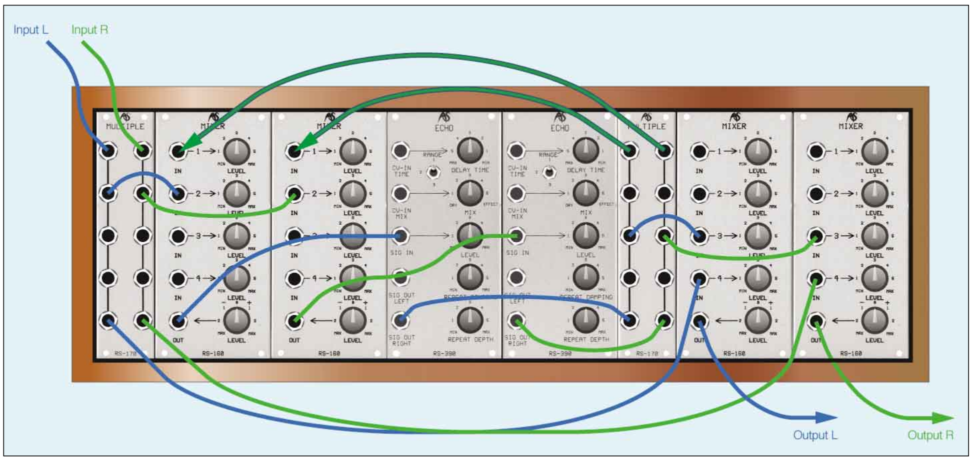


Figure 12: Patching the 'ping-pong' delay.

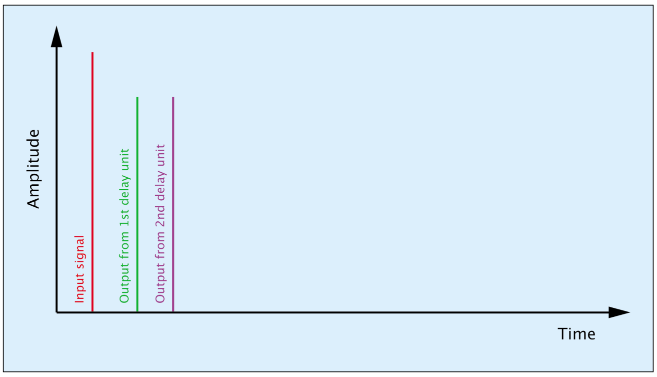


Figure 13: Two delays without regeneration.

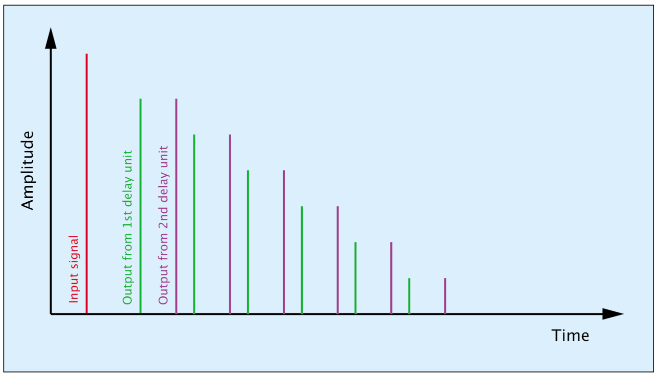


Figure 14: Using Delay Line 2 to echo a series of echoes produced by Delay Line 1.

for the right channel, and blue/green for the two that form the cross-feedback paths) you'll see that it makes sense. What's more, it's clear that you now have independent control over every aspect of the effect, so you could have different delay times, intensities and levels for the two channels, leading to even more complex sounds. And we can go still further...

## Primitive Electronic Reverb

Despite their power in certain areas, tape delays and simple delay lines suffer from a significant limitation: each delay line produces a single, discrete echo. So the complexity of the resulting sound is, well... not complex at all. If there is no regeneration in the feedback loop, we can represent the output from two delay lines — say, the first two in Figure 9 — as shown in Figure 13 (above).

But the RS390 is *not* a simple delay line, and there are plenty of more complex alternatives, particularly in software-based synths, that allow us to be more ambitious. In my patch, the first RS390 alone is capable of producing a string of delays and then presenting them to the second RS390 (we get it to do so by increasing the Repeat Depth by a suitable amount). If we now consider the output of this, we obtain something that looks like Figure 14 (above right).

This is more pleasing, but still sounds like a discrete sequence of echoes. So let's now set up the second delay unit to create a stream of echoes, just like the first. To make the result interesting, we need to set the delay times on each unit differently, or all the echoes will fall at the same time. If the second unit is repeating slightly more quickly than the first, the result looks like Figure 15 (below).

Now comes the final touch, as we introduce a third delay unit, or, in my patch, the third RS390, set to an even shorter delay time than the previous two, and set to produce a stream of echoes from each of the echoes in Figure 14. The result, drawn in Figure 16 (below), looks suspiciously like the reverberation in Figure 18 (on the next page), which I've copied unmodified from the February 2001 instalment of Synth Secrets, where we discussed reverb in some detail. Indeed, if I recolour Figure 16 to illustrate the division between the early reflections and reverb tail (see Figure 17, also on the next page) you can see that the relationships are preserved remarkably accurately. OK, I'll admit that the result lacks some of the qualities of true reverberation, and that it sounds somewhat 'electronic', but it would definitely sound like reverb to most people, and would not be perceived as a complex set of echoes.

Yet this isn't the end of the story, because if the delays in this patch are short enough for the early reflections to sound realistic, and for the echoes to be dense enough to sound reverberant, the tail in Figure 18 will be rather more 'bathroom' than 'Grand Canyon'. So now we invoke the patch-cord regeneration loop in Figure 9, sending the whole thing round and round to become ever more complex as it decays. The echoes soon turn into a cloud of thousands or even millions of indistinguishable repeats, which is, of course, exactly what natural reverb is. Nonetheless, the effect still exhibits a somewhat 'metallic', sound because the delay times in each RS390 are constant, so there are three, enharmonic, characteristic frequencies present. The same happens in spring reverb, where the longitudinal, latitudinal and torsional vibrations provide three modes of vibration, which is why spring reverbs have that metallic ring to them. But the results can be extremely useful as novel effects rather than as precise recreations of acoustic spaces.

If you have an opportunity to configure a set of delays in this way, you'll find that you can play with the Repeat Depth and Damping of each delay line to create many reverberant effects. Just be careful not to create an unstable system that attempts to remove the cones from your studio monitors...

If you're lucky enough to be using delay units that possess dual left/right outputs (as do the RS390s I've used this month), you can

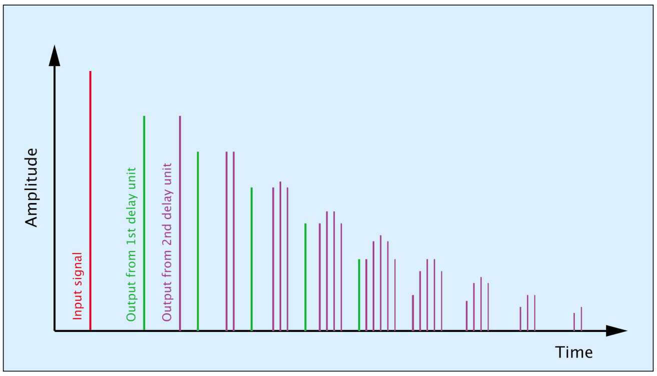


Figure 15: Producing a stream of echoes for each of a stream of echoes.

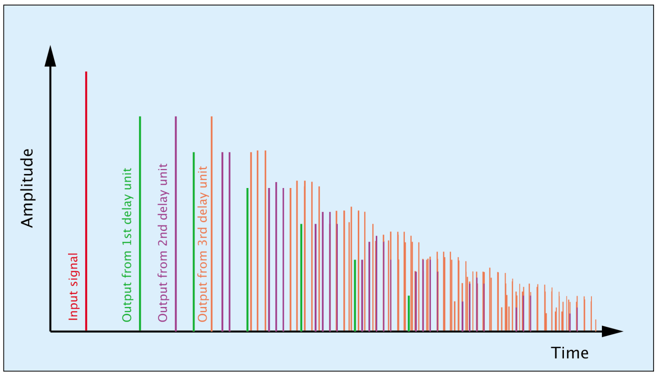


Figure 16: Producing even denser streams of echoes using all three delay lines.

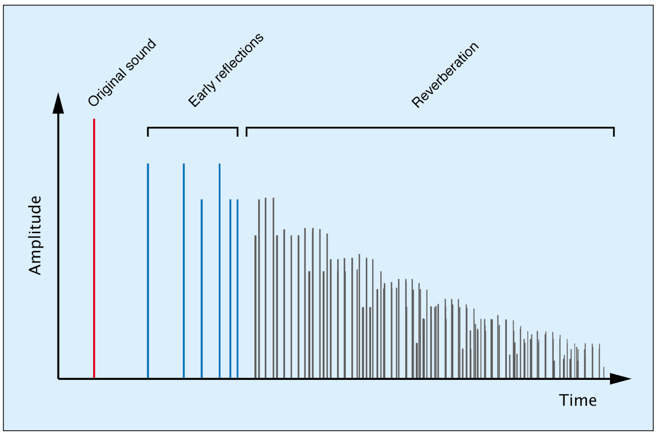


Figure 17: Creating reverb using three delay units with regeneration.

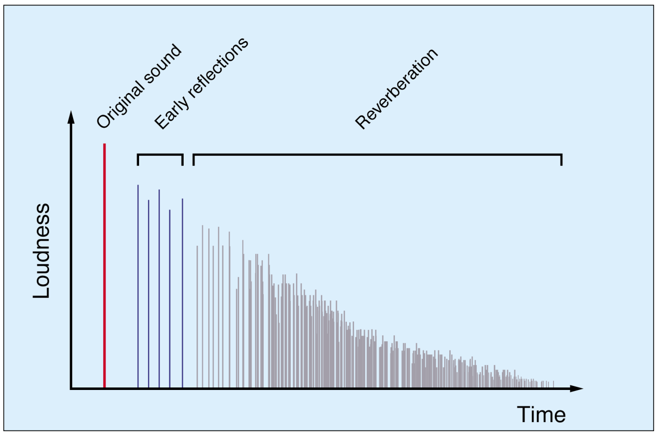


Figure 18: A representation of natural reverberation, from February 2001's instalment of Synth Secrets.

increase the reality of the effects still further. This is because natural reverberation in a room produces subtly different results at each of your ears. If it did not, it would be like listening to a monophonic recording through a stereo system; despite the fact that the sound is coming from two sources, there is no feeling of spaciousness. This is also one of the reasons why concert halls have high ceilings... they ensure that the early delays do not reach each ear simultaneously by way of the ceiling, but at different times by way of the walls. Anyway, by adding a couple of additional mixers and repatching the RS390s in Figure 9 to take advantage of the dual outputs, you can transform what started out as a simple recreation of a three-head tape delay into a remarkable stereo reverb unit. Don't you just love this stuff?

## Epilogue

Before we finish looking at fun things to do with delay lines, we must find out what happens when we start to modulate their delay times. I'll be giving no secrets away if I tell you that this moves us firmly into the territory occupied by choruses, flangers, and ensemble effects... so that's where we're going next month. Until then...

# Synth Secrets

## More Creative Synthesis With Delays

**In the penultimate instalment of this long-running series, we delve deeper into what can be achieved with just a few delays and some creative routing...**

Gordon Reid

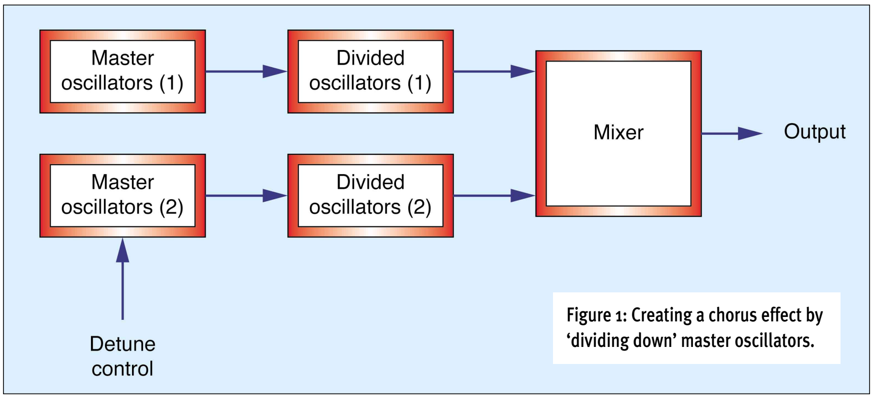


Figure 1: Creating a chorus effect by 'dividing down' master oscillators.

Here we are, at the far end of our synthesizer's signal path. We've generated the waveforms, created multiple signals, filtered and sculpted them, applied modulation, mixed the results, and... well, it all sounds a bit *thin*, doesn't it? Despite the techniques we've employed, the results totally lack the depth of a nine-foot six-inch Bösendorfer, or a four-manual cathedral organ. Yet, if you think back to the early 1960s, there was — apart from a choir or the string section of an orchestra — little that was musically lush, and the electronic sounds that we now take for granted were still in the future. If you wanted to record the semblance of multiple instruments playing at once, you either paid a few dozen instrumentalists to do their thing simultaneously, or you bought a MkII Mellotron. Either way, the costs were crippling.

Things began to change in the mid-1960s, when the affordable 'chorused' organ was born. Consider the way in which a cheap electric organ creates its sound. In general, the outputs of high-frequency oscillators are 'divided down' by integer factors to create the correct pitches for all the notes of the top octave of the keyboard, and these are then further divided by factors of two to generate each octave beneath. However, organ designers discovered that they could divide the master oscillators in different ways to generate two frequencies for each note that

▶

### Creating Chorused Sounds Without Chorus

**If a polyphonic synth has dual oscillators (or better still, three) per voice, it will be capable of creating thick, quasi-chorused sounds, even without a chorus unit.**

**To create these sounds, you must first select the sawtooth wave option on one of the oscillators (because this has the correct harmonic content for a string sound) and the pulse wave on the other. Secondly, you must detune one oscillator against the other to create a 'detuned' sound. Next, you must add pulse-width modulation (or PWM) to the pulse wave.**

**As I showed in the March 2003 instalment of this series (see www.soundonsound.com/sos/mar03/articles/synthsecrets47.asp) pulse-width modulation generates two 'virtual' signals, with one being pitch modulated with respect to the other (see Figure A, right). PWM alone creates a 'chorused' timbre, but if you detune the sawtooth wave with respect to the pulse-width modulated wave, there are, in effect, three pitches present in the output, and this further thickens the sound (see Figure B). Finally, adding vibrato to the sawtooth wave will complicate the relationships between the three pitches, especially if the synth can modulate the vibrato and PWM at different rates (see Figure C).**

**The key here is to ensure that there is so much activity that your ear becomes unable to recognise the limited number of pitches present. When programmed carefully, the sounds produced by this method can be superb, as evidenced by the remarkable 'ensemble' patches produced by the Prophet 5, OBX and, in particular, the Roland Jupiter 8.**

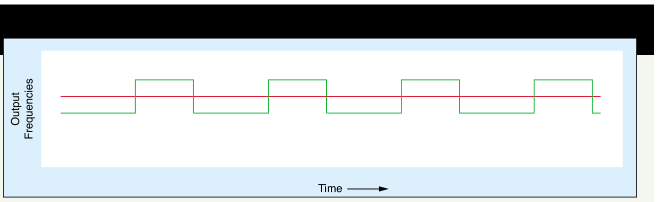


Figure A: Thickening the sound by modulating the duty cycle of a pulse wave.

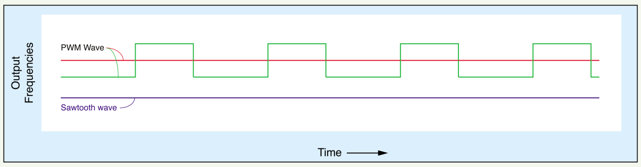


Figure B: Adding another signal to further thicken the sound.

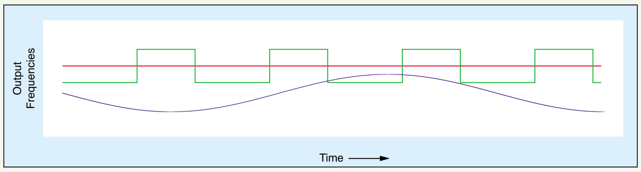


Figure C: Creating a lush sound using two oscillators per voice.

were almost, but *not exactly* the same.

These small discrepancies — which were not even identical from note to note — were not dissimilar to the differences between the pitches of two pipes tuned to the same pitch, or between the three strings that comprise a note on the aforementioned Bösendorfer. Consequently, manufacturers began producing electric organs that generated a primitive chorus effect using dual sets of dividers. More sophisticated designs incorporated two independent sets of master oscillators, each with a set of frequency dividers, and some even offered a global detune for the second set (see Figure 1, on the previous page).

Of course, there was also the principle of using multiple electronic sound sources playing in unison to recreate the effect of multiple physical sound sources playing in unison. So, long before the appearance of the modern chorus effect in 1975, keyboards were using multiple oscillators per note to thicken up what would otherwise have been bland and uninteresting patches. Of these, the most sophisticated was the prototype of Ken Freeman's String Symphoniser. This used three banks of detuned oscillators, applying vibrato to each to create a rich chorus effect. To this day, synth programmers often use detune, pulse-width modulation and frequency modulation to obtain richer timbres than would otherwise be possible (see the box on the previous page).

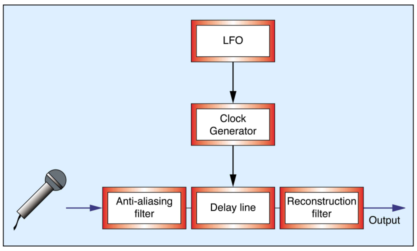


Figure 2: A modulated delay line.

Nevertheless, this is not what we now mean when we use the word 'chorus' and, of course, it can't be applied to externally generated signals such as a human voice or a note played on a guitar. What's more, despite the complexity of detune/vibrato/PWM programming, timbres generated in this fashion still don't sound as lush as even the cheapest and cheesiest 'string synths'. On the surface, this is rather surprising. In contrast to a sophisticated multiple-oscillator-per-note synth, the initial waveform produced within a string synth is almost always a single, 'divide-down' sawtooth which, at the best of times, sounds weedy and uninspiring. Yet, passed through the instrument's internal chorus/ensemble unit, it sounds animated, lush and full of body. So... what is chorus, and why does it sound so good? To answer that, and before Synth Secrets departs through the output socket of history, it's time that we took a look at that most popular and most useful of all keyboard effects: the chorus/ensemble.

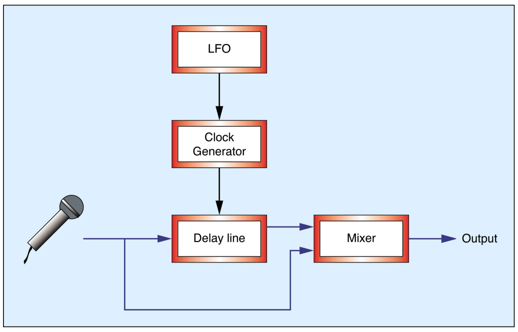


Figure 3: Adding the unaffected signal to the pitch-modulated signal.

## A Basic Chorus Effect

The secret to this effect is fooling the ear into thinking that it is hearing multiple performances of the same note when it is not. This may sound tricky, but the key to doing so already exists in my explanation of modulated BBD delay lines, which I introduced in March's instalment of this series (see www.soundonsound.com/sos/mar04/articles/synthsecrets.htm). In short, if you don't have access to multiple, closely related timbres and pitches, why not split a single signal into multiple paths, apply pitch modulation to one or more of these, and then remix them? It's a simple idea, and it works beautifully.

Figure 2 (left) shows the structure of a modulated delay line. You can present any signal to the input, whereupon it will be low-pass filtered to eliminate aliasing, sliced into samples, and then passed through the line before being reconstructed at the far end.

This, by the way, is as true for a digital delay as it is for an analogue BBD. The speed at which the samples travel down the line, and their precise temporal relationships (ie. how far apart they are spaced) are determined by the clock generator and the oscillator modulating its speed. I hope that it is obvious that, if the clock is running faster when a bunch of samples reach the end of the line than it was when they entered, the samples will be closer together, so the pitch will be higher. Conversely, if the clock is running slower when those samples reach the end of the line, the samples will be further apart, so the pitch will be lower. Clearly, this allows us to modulate the pitch of the signal, and if the LFO in Figure 2 were generating a sine wave, the output from this diagram would exhibit a pronounced 'wow' effect, like a vinyl record with the hole punched in the wrong place.

Now, let's add a second signal path to Figure 2, which allows the unaffected signal to pass to the output, as shown in Figure 3 above (from which, for clarity, I've omitted the anti-aliasing and reconstruction filters). We now have the situation where the modulated signal is sometimes at a higher pitch than the 'straight-through' signal, sometimes at a lower pitch and, on two occasions in each cycle, at the same pitch. Figures 4 and 5 (below) demonstrate this, showing how the delay in the upper signal path changes in time, and how this affects the pitch relationships between the upper and lower signal paths.

We can patch Figure 3 very easily using just four modules from a modular synth: a multiple to split the incoming signal into

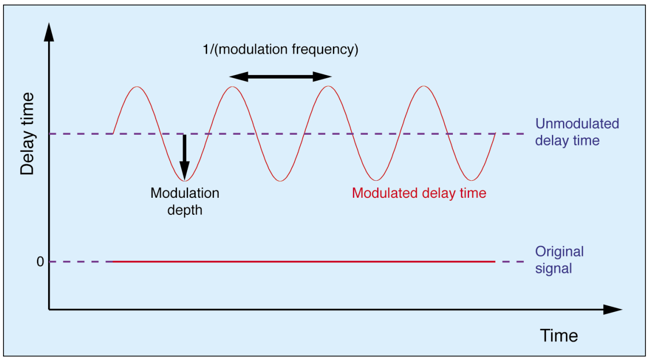


Figure 4: A simple delay modulation.

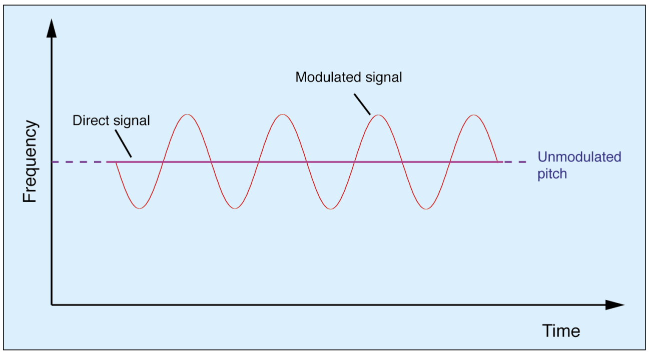


Figure 5: The pitch-shifts resulting from the modulation in Figure 4.

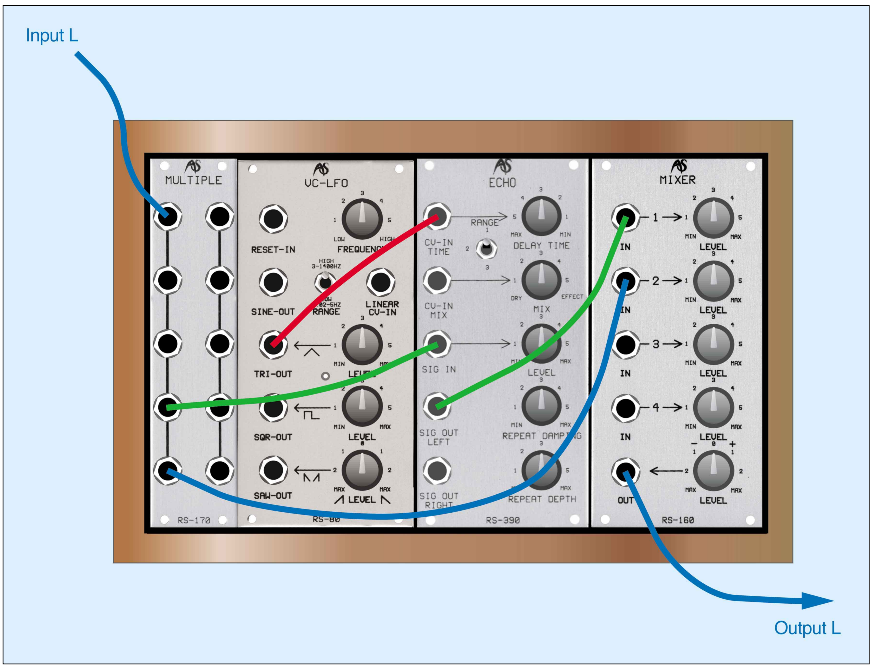


Figure 6: A simple, two-path chorus unit.

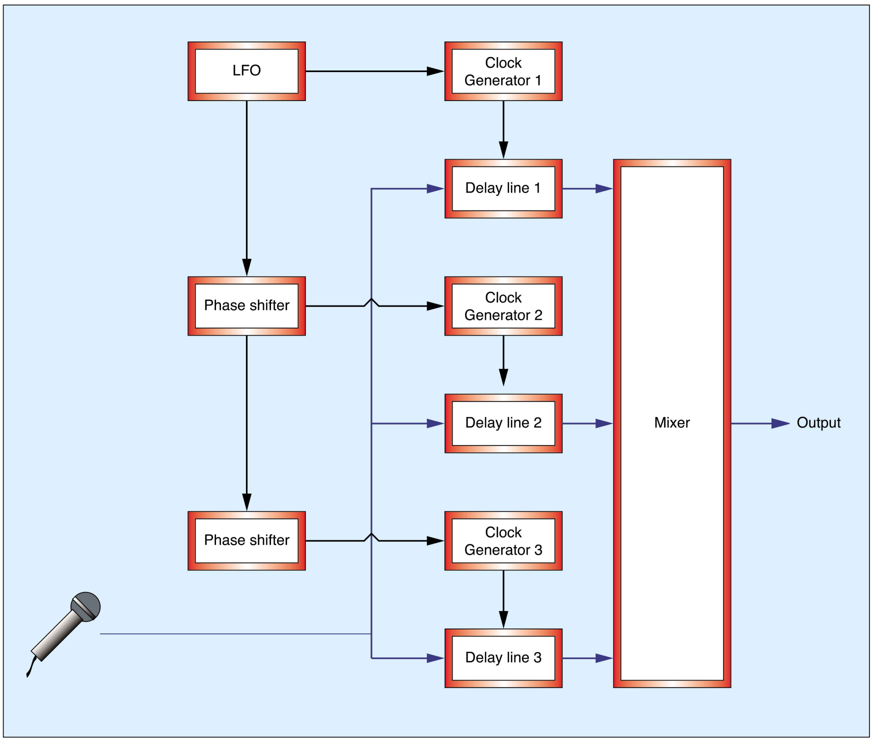


Figure 7: Using a single LFO to modulate three delay lines.

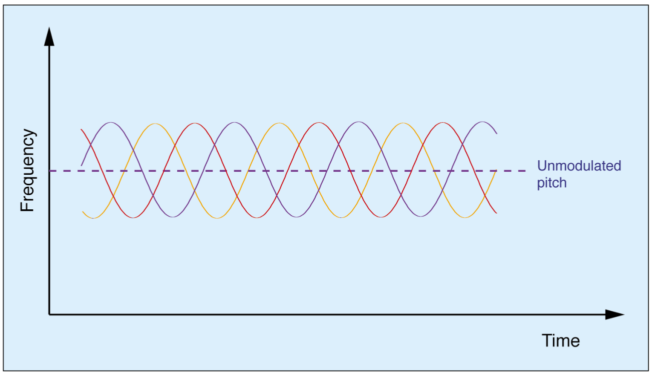


Figure 8: The frequency modulations of the three signal paths in Figure 7.

▶ two paths; an LFO to modulate the rate of the Echo unit (which combines the delay line and clock generator in a single module); and a Mixer to recombine the two audio signals. In Figure 6 (left), I have shown the original signal and output in blue, the modulation path in green, and the modulation signal in red, just to make things clearer.

Of course, this patch will produce nothing like the desired effect unless we choose our parameters sensibly. Firstly, you'll find that the barest minimum of modulation is needed. If you can't obtain a low enough modulation level from the Level knob on the LFO, place an attenuator in the modulation signal path to reduce the amplitude even further. In contrast, you have a wide choice of modulation rates. A slow sweep at a fraction of 1Hz will provide a gentle chorus, while a faster rate — say, 5Hz to 7Hz — will result in a more typical 'synth' ensemble. The other vital factor is the delay time. Set this to be too long, and you'll hear a distinct delay. Set it too short, and you'll obtain a version of another effect: flanging. But get it right — somewhere in the range 10ms to 50ms, as your taste dictates — and then mix the two signal paths in equal measure, and you'll obtain a serviceable chorus, reminiscent of the cheapest '70s string synths. Hang on... the *cheapest* '70s string synths? The individual modules in this patch (or an equivalent software modular synthesis application) could cost a couple of hundred quid, so you've a right to expect something a bit better. What's going on?

Unfortunately for us, our ears — which evolved to locate the rumbling tummy of a sabre-toothed tiger at 500 paces — are not fooled by the two signals generated and mixed in Figures 3 to 6, so this implementation of chorus is not perceived as particularly deep, nor indeed as particularly lush. In effect, it's the equivalent of hearing just two singers, or just two violinists, when what you're after is the choir and orchestra performing Mahler's Ninth in the Royal Albert Hall. So we overcome the problem by adding more signal paths to the existing scheme, and by modulating all of them differently.

I have shown an efficient way to do this in Figure 7 (above left), which shows that we can use a single LFO to modulate each of the delays, provided that the phase of each instance of the LFO waveform is shifted by some amount. Without these shifts, the three audio paths would be modulated identically, and we would create vibrato, nothing more. If the three paths are modulated at relative phases of 0 degrees, 120 degrees and 240 degrees, we obtain the pitch relationships shown in Figure 8 (left). As you can imagine, this is a far more complex ▶

sound, and the relationships between the three modulated signals provide a thicker and warmer chorus effect.

## Improved Chorus Effects

The chorus described in Figures 7 and 8 is a classic '70s design, and was used in numerous string ensemble keyboards, but it still does not have the richness and depth that we have come to associate with the best of such effects. This is because synthesizer designers continued to dream up better ways to modulate the input signal, the first of which was to modulate each delay line with not one LFO, but two.

This was conceptually simple, although debate raged over the speed and depth that creates the most pleasing effect. Many manufacturers opted for two very different speeds — one of the order 0.5Hz to 0.7Hz, with a second closer to 6Hz to 7Hz, but with no discerned integer relationship between the two, so that the modulation didn't repeat for a long time. Figure 9 (left) shows two such waves, and the resulting modulation signal. As you can appreciate, it's going to be more difficult for Ugg the Caveman to perceive this as a simple, repeating waveform.

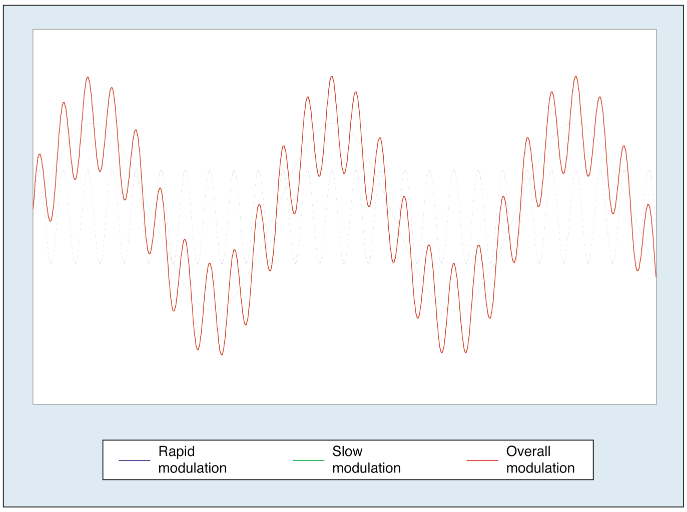


Figure 9: Creating a more complex modulation from two sine waves.

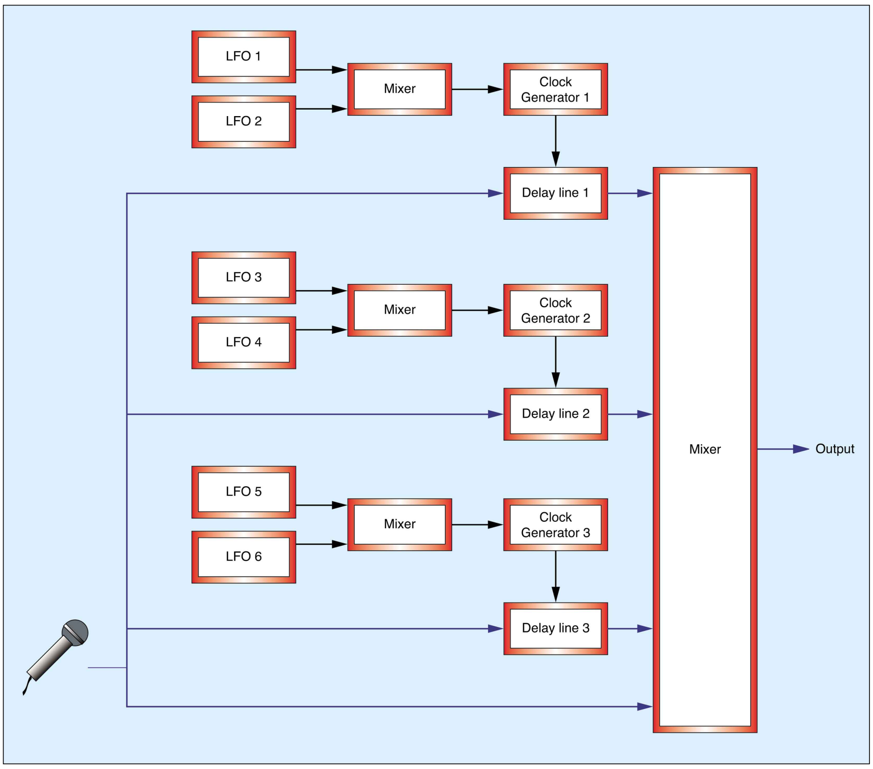


Figure 10: A four-path chorus unit.

Now consider Figure 10 (below). This shows the input split into four paths — the original signal plus three delayed versions of it. Each of the six LFOs in the patch can have a different modulation rate, and each of the mixers in the modulation paths can be designed so that each of the LFOs contributes a different depth. The result is a lush, complex swirl of sound that is forever evolving, and which adds movement and texture to even the most basic of initial waveforms.

Another variation on this approach rearranges the LFOs so that one of each pair modulates the frequency of the other. This arrangement, which I've shown in Figure 11 (bottom) produces the waveform in Figure 12 (on the next page), creating yet another form of pitch modulation and, therefore, a subtly different chorus. One could go even further, for example using another LFO to modulate the depth of the modulating waveform as well as its frequency and, if cost were no object, you could keep slinging LFOs, mixers and, where necessary, VCAs at the problem to create the most complex modulations imaginable. You can even use a random waveform as a modulator, which goes some way towards imitating the genuine pitch instabilities of human singers and players.

As you might imagine, circuits such as Figure 10 can be expensive to build, and although this design produces a superb ensemble sound, it may not be economical. To overcome this, many manufacturers combined the ideas set out in Figures 7 and 10, employing a trick that fools the ear into believing that it's hearing multiple, complex modulations, when in fact only one is present (see Figure 13, on the next page). This involves the use of just two LFOs (which cuts costs) and four phase-shifters (which are cheap), and generates three instances of a single complex delay modulation. As before, these are out of phase with one another, typically by 120 degrees, and the result, while not quite as lush as you can obtain using six independent LFOs, is nonetheless gorgeous. This is why this method — or close variations of it — became the standard for almost all the best-loved chorus/ensemble keyboards.

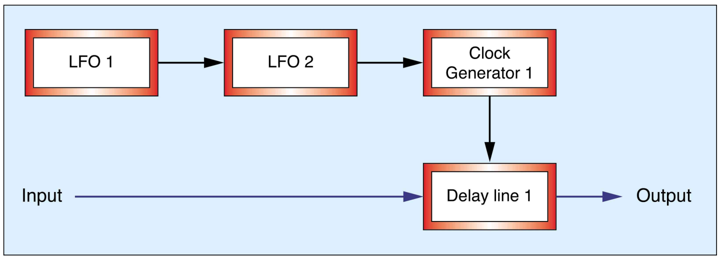


Figure 11: Reconfiguring the LFOs to produce a modulation signal with vibrato.

## Stereo Chorus

Nice as monophonic chorus might be (and is), it doesn't make the most of the techniques described above. Indeed, when Roland first

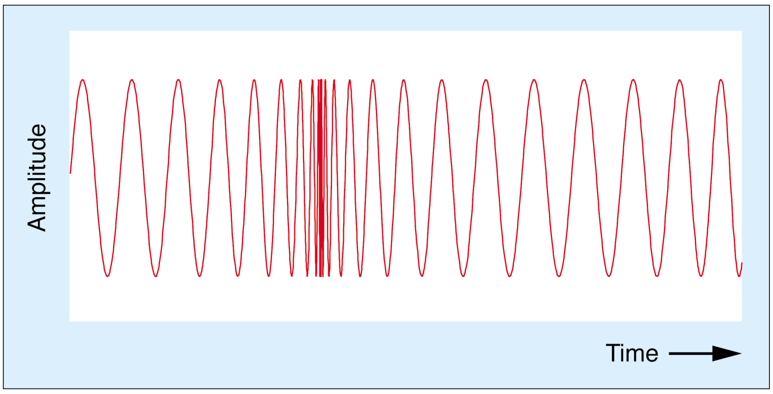


Figure 12: Modulating the speed of the modulator.

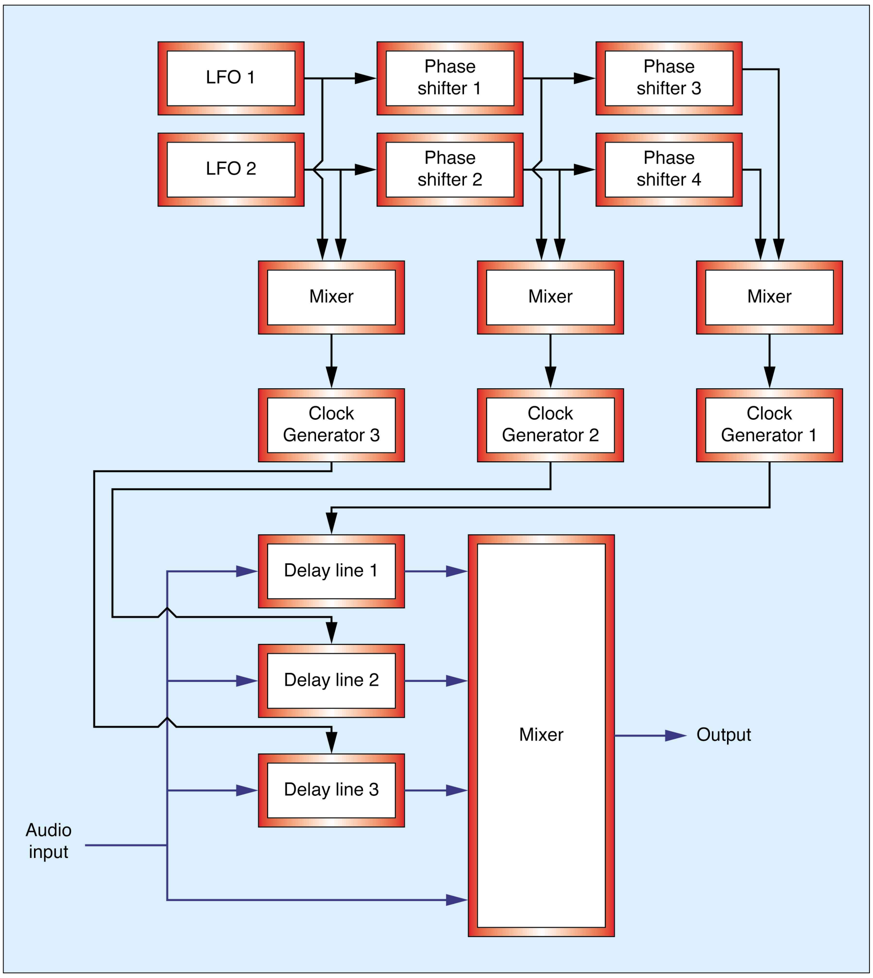


Figure 13: The classic three-phase chorus unit.

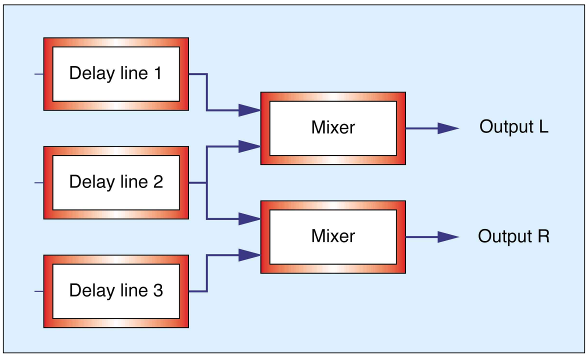


Figure 14: The output stage of a triple-path stereo chorus unit.

designed the chorus/ensemble pedal, they created a CE1 mountain, because nobody bought the things. It wasn't until a handful of players discovered the 'Stereo' output (which sits less than an inch to the right of the Left/Mono output) and directed this to a second amplifier and speaker that the world sat up and started to take notice.

As we now appreciate, stereo chorus is *the* classic synthesizer effect, and it's easily created if you have access to the signals being produced by the multiple delay lines. Figure 14 (bottom) shows a triple-path configuration in which the signals generated by the first and second delays are mixed and sent to the left output, while the signals generated by the second and third delays are mixed and sent to the right. In this scheme, you can leave out the 'straight-through' signal, because the dual inputs to each mixer will be chorusing differently, and — far from contributing to the result — the original might actually damage the impression of width and depth.

## Some Real Examples

Although some people think that string synths sound much like one another, this is not true. Sure, they all share the same *class* of sound, they are almost without exception based upon quasi-sawtooth waves generated by organ-style 'divide-down' technology, and if you switch off their chorus/ensemble effects, they are all about as interesting as a bunch of boring things on a very boring day. Nonetheless, there are marked differences in the way that they generate their chorus effects.

For example, the Eminent Solina (1974) uses the chorus structure depicted in Figure 11, with two LFOs, one running at around 1Hz and the other at about 6Hz, phase-shifted to produce modulating signals at 0 degrees, 120 degrees  and 240 degrees. In contrast, the Roland VP330 (1978) has a thinner string ensemble sound generated by just two delay lines with dual LFOs. The altogether richer-sounding Korg Polysix synthesizer (1981) incorporates another triple-delay chorus, but dispenses with the phase-shifters and uses a configuration of independent LFOs similar to that shown in Figure 10.

One of the cleverest of chorus designs was developed by Roland (see Figure 15, on the next page). This uses just a single square wave LFO (generated, would you believe, by one of the flip-flops that I described a couple of months ago) and three frequency-dividers that output modulation signals at the clock frequency $F_c$, at 1/2 $F_c$, 1/4 $F_c$ and at 1/8 $F_c$. These 'square' signals are low-pass filtered to 'round off' the waveforms into approximations of sine waves, and these are in turn used to modulate the clocks driving four BBDs. The outputs of the delay lines are then mixed into

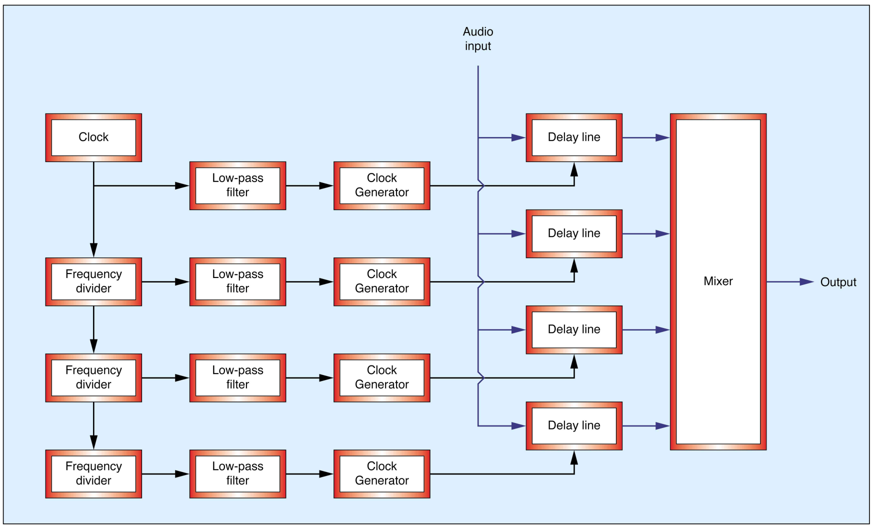


Figure 15: A 1978 chorus design by Roland Corporation.

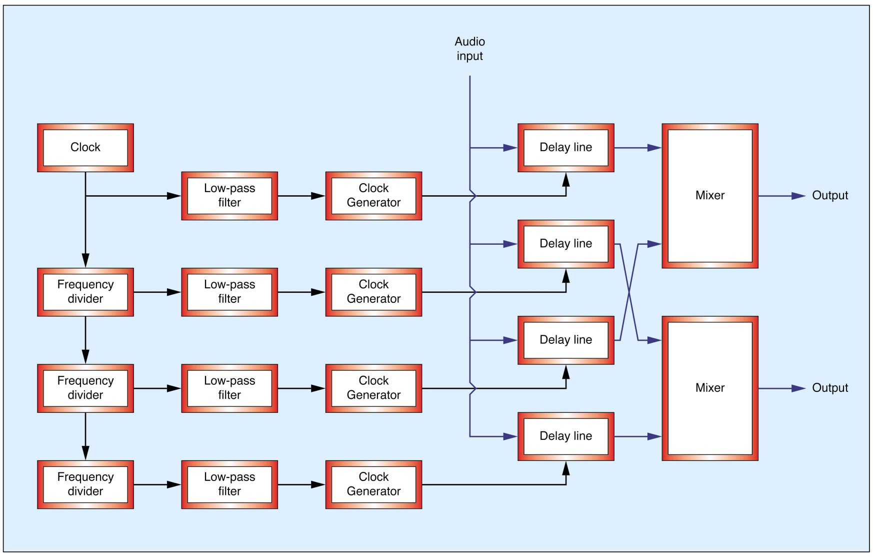


Figure 16: Converting the Roland chorus in Figure 15 into a stereo unit.

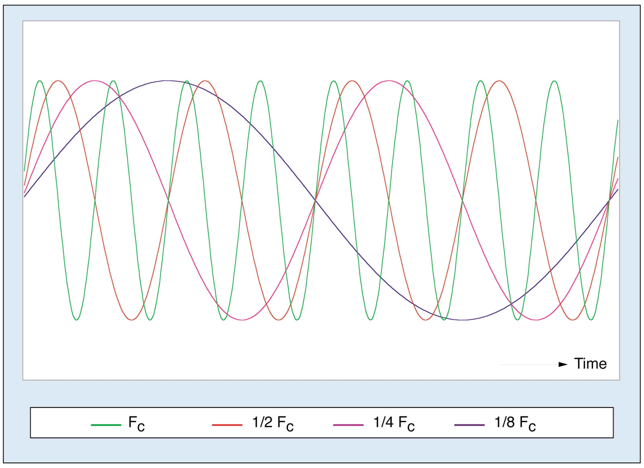


Figure 17: The modulations generated by the clock and dividers in Figure 15.

a single sound, and emerge in glorious mono. There is no 'straight-through' signal present in the output.

The major advantage of this configuration is one of cost; clocks, dividers and simple low-pass filters are cheap. However, there is an additional benefit, which was the true *raison d'être* of the design: due to the nature and relationships of the four modulating waves (shown in Figure 17, left) the chorus effect is less susceptible to the 'warbling' effect that you hear generated by other low-cost circuits.

Roland even suggested ways in which Figure 15 could be improved. For example, you could mix the outputs from the first and third BBDs and from the second and fourth, to create a stereo chorus (see Figure 16, above). The designer also suggested inverting the phase of the second and fourth modulating waves to create an even richer 'spread' of the ensemble effect, and using frequency dividers that used other integer factors of $F_C$, such as one third or one fifth. Yet another Roland design dispensed with sine-wave modulators and substituted sawtooth waves, eliminating the pitch discontinuities by modulating a set of VCAs that silenced the delay lines while the modulating waves reset to the start of their cycles. Yet another incorporated signal gates that disconnected the modulation waveforms when there was no audio signal present at the input, thus eliminating the characteristic 'swishing' noise that mars many chorus effects. The permutations are almost endless.

## Epilogue

Despite the obvious benefits of chorus units, BBDs are rather noisy, so many (although not all) 1970s manufacturers treated them as a necessary evil that added interest to cheap, single-oscillator keyboards and synthesizers, but which were not suitable for top-of-the-range instruments. But while the high-brow approach was good in principle, the synthesizers that eschewed chorus and relied on other techniques to create lush sounds were expensive: multiple VCOs per voice, the ability to detune independent banks of oscillators against one another, the ability to modulate the pulse width of one bank and the frequency of the other... it all added up.

Then, sometime in the 1980s, chorus effects became legitimate, and started to appear on multi-oscillator-per-voice synths as well as single-oscillator instruments. Ask yourself, what's the thing that most synth fans mention first when discussing the Elka Synthex? The oscillators? The filters? The modulation? No... it's the superb chorus unit at the end of the signal path. Even Sequential Circuits conceded defeat in the late '80s, by adding chorus to the Prophet VS. And what do people most often decry about some of the most powerful 'pure' synthesizers yet developed? It's that they didn't sound lush, so Yamaha redesigned the DX series, restoring the chorus unit that they had removed when they discontinued the GS1 and GS2 in 1983 or thereabouts.

By the late '80s, it had become clear to everyone that you could take the less animated sounds from DCO-based synthesizers, or the relatively sterile waveforms from early digital synthesizers and workstations, add a well-designed chorus unit, and the thing would sound gorgeous. Hmm... a bland waveform enlivened by a chorus/ensemble... we used to call that a string synth, and it's one of the enduring absurdities of our industry that instruments such as the Solina and ARP Omni now sell for more than double the price of a well-preserved DX7. SOS

# Synth Secrets

## The Secret Of The Big Red Button

Gordon Reid

**After over five years, Synth Secrets reaches its conclusion (and conclusions!). Will we ever look at synthesis in quite the same way again?**

Every few weeks, my colleagues at *Sound On Sound* and I receive a telephone call or email demanding to know which is the best synthesizer. When these requests come directly to me, I have often replied, asking the enquirer what he (it's invariably a he) wants to achieve, and what criteria he deems to contribute to the term 'best'. And, every time I do, the response is, "Look, never mind all that... just tell me which one is best. I know *you* know — why won't you tell me?"

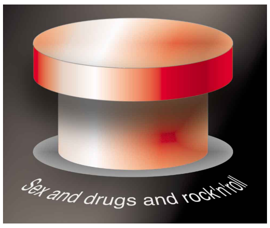


Figure 1: The Big Red Button — surely it's in there somewhere?

Sometimes, I have tried to help the caller by asking him to think about what is most important to him — huge polyphony, sequencing, complex arpeggiation, suitability for live use, ethereal floaty sounds, heavy bass, rhythm sections, and so on. And the reply is invariably, "Look, never mind all that... I'm not interested in all that technology. Just tell me which one is best. I know *you* know — why won't you tell me?"

You might think that this attitude is rare, but it isn't. If anything, it's becoming more common, perhaps because manufacturers are determined to convince you that you will become the latest, biggest, and richest musical phenomenon the world has ever seen if you simply buy the latest version of their Argon Megastation with the added wotsit and optional thingies.

There's even a second stage to this delusion. It's the feeling that, hidden somewhere on the Argon Megastation, there's a Big Red Button marked 'Number 1 hits, free drugs, and more sex than you can handle'. Few musicians admit to searching for the Big Red Button, but many believe that it exists, and some are really sore that I won't tell them where it is.

### The Secret Of Synth Secrets

Somewhat over five years ago, the very nice people at *Sound On Sound* and I were discussing the belief in the elusive Big Red Button, and we agreed that it would be interesting to publish a series that explained some of the less frequently discussed aspects of sound and synthesis. The idea was to avoid the introductory tenor favoured by other such series (your VCO-bone's connected to your VCF-bone, your VCF-bone's connected to your VCA-bone... and so on) and to tell you things that you didn't even *know* you wanted to know. Even at the start, it was clear that no single approach would be suitable for all readers. One respondent described the first two parts of the series as 'condescending', even comparing them to a nursery rhyme, while others simultaneously claimed that the same parts were far too intense and mathematical. Nonetheless, over the ensuing five years, we have covered more ground than any previous *SOS* series on sound generation and synthesis.

But now I find myself just a handful of paragraphs from the end of the final part, and I would like to leave you with one overriding idea that, in my opinion, sums up everything we've discussed. I hope that this will help you to become a better sound designer and player, although this seems a tall order for just one article.

To attempt this, I'm going to offer you an alternative to the manner in which most people approach their synths. I call this 'modular synthesis', but you won't need a £10,000 wall of vintage modular synthesizer modules to take advantage of it.

### A Different Way Of Thinking

Many books on synthesis begin by explaining that sounds can be characterised at each point in time by three qualities: pitch, timbre and loudness. This idea is then extended to state that you can partition most synthesizers into oscillators that determine the pitch, filters that determine the timbre, and amplifiers that determine the loudness. Just throw a couple of contour generators and low-frequency oscillators into this brew and you have (so some would say) everything that you need to synthesize every sound imaginable (see Figure 2 below). However, I find this approach to be rather limiting, so, instead of viewing synthesizers in this way, I'm going to propose that you consider them in terms of three

▶

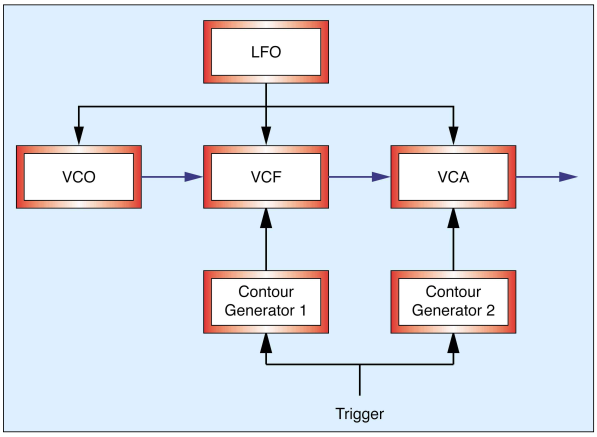


Figure 2: A simple synthesizer architecture.

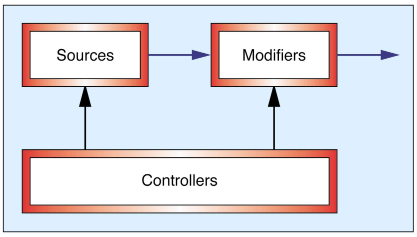


Figure 3: Thinking about the synthesizer in a different way.

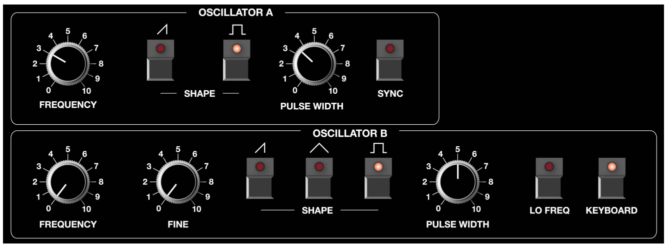


Figure 4: The Prophet 10 oscillator section offers a range of common waveforms.

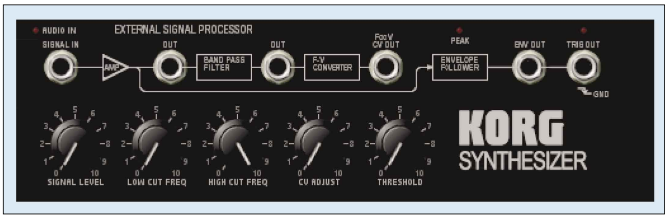


Figure 5: The external signal processor (ESP) from the Korg MS20.

major classes that I call Sources, Modifiers, and Controllers (see Figure 3, above).

## Sources

The most common sources on synthesizers are audio-frequency oscillators (such as those shown in Figure 4, above) although there are others, including noise generators and the external signal inputs offered by, for example, the Korg MS20's ESP section (see Figure 5, above).

The range and quality of the sources in your synth will place tight constraints upon the sounds that you can create. Clearly, you can't easily synthesize a hollow sound without an initial waveform that has that characteristic, nor can you create brassy sounds without waves (or, for that matter, samples) that offer the appropriate harmonic structure.

So it's a good idea to learn to recognise the sounds of various waveforms, especially the ones commonly found on synths, such as sawtooth, square, pulse, and triangle waves. If you can hear a sound — whether produced acoustically or on another synth — and judge the waveform or combination of waveforms that is most appropriate to emulate it, you've taken a huge step toward earning the title of synthesist. Likewise, if you can imagine a sound, and choose the right waves to make it a reality, you're well on the way to becoming a sound designer.

Of course, it becomes harder to identify waveforms as the patches that contain them become more complex. You may think that if you envelope them, pass them through a filter or two and apply various forms of modulation, it would become impossible to identify the waves themselves. But you might be surprised. The ear/brain combination is a remarkable tool, and once you have learned to recognise waveforms in isolation, you'll be amazed at your ability to identify them in more difficult circumstances.

## Modifiers

Modifiers are those parts of the synthesizer that modify the signals produced by other parts of the synth. Most obviously, these are the filters and amplifiers illustrated in Figure 6 (below). Unfortunately, many synth users seem to have become obsessed with the class of modifier known as the low-pass filter. Every discussion about synths seems to centre on similar questions: 'What filter does it have?' or 'What is its cutoff slope?'

To be fair, the characteristic sound of a synth's filters *is* a major arbiter in the sounds that you can obtain from it. Nonetheless, we should not concentrate only on low-pass filters, because there are many other modifiers. High-pass filters, band-pass filters, band-reject filters and comb filters all have their place in shaping the timbre of your signals. Less obvious modifiers include sample & hold generators, ring modulators, frequency-shifters, slew generators, reverbs, and even chorus units, all of which have been described in detail during this series. There are no rules that say where these should lie in the signal chain, and if you want to modify your audio signals using, say, reverb and chorus *before* any form of filtering or loudness shaping, there are plenty of synths that will let you do so, especially affordable software-based ones.

## Controllers

The third major class in this scheme is that of the 'controllers'. These generate the signals that determine how the other parts of the synthesizer are operating. Again,

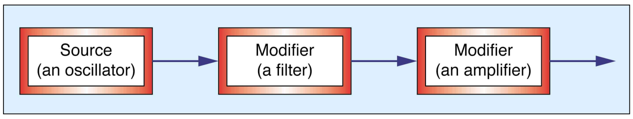


Figure 6: Thinking in terms of sources and modifiers.

## Adding A Bit Of Interest

**Even if you have a clear idea how to use the building blocks of your synthesizers, there are still a few other things it's a good idea to consider when designing and playing sounds. For example, it's sensible to think about timing signals, and be clear about the differences between triggers and gates. You should also make sure that you know when it might be preferable to use single-triggering, multi-triggering, and the various types of key priority, as well as when monophony is superior to polyphony. You'll find all of these discussed if you look back over the course of this series.**

**But not everything is cerebral. Just as important as your understanding of sounds is your ability to play them. If you want to sound good, it makes sense to develop good techniques for using controllers such as pitch-bend and modulation wheels, joysticks and aftertouch. Unfortunately, the number of synthesists who ignore articulation and expression has led to an oft-quoted statement, usually made by guitarists of the teeth-clenching, groin-thrusting variety, who believe that wringing the neck of a bit of dead tree constitutes a purer form of music. They claim that "synthesizers are just a big collection of soulless switches."**

**They're wrong, of course, but the inability of many keyboard players to coax even a modicum of expression from a synth lends credibility to their views. You can't solely rely on contour and modulation generators to do the hard work for you. While their great strength is that they make everything consistent and repeatable, so that a sound is recognisably the same from one note to the next, their great weakness is that they make everything consistent and repeatable, so that every note sounds the same from one to the next.**

**So, next time you're making music, why not try using the physical controllers on your synth to adjust the way in which a note 'speaks', or add expressive vibrato (or tremolo, or growl) using the pitch-bend wheel or ribbon, rather than (say) relying on an LFO which always produces the same effect. It doesn't matter whether you're playing in an orchestral style, or prog-rock, or dance, or industrial techno… you might be surprised at the possibilities this opens up.**

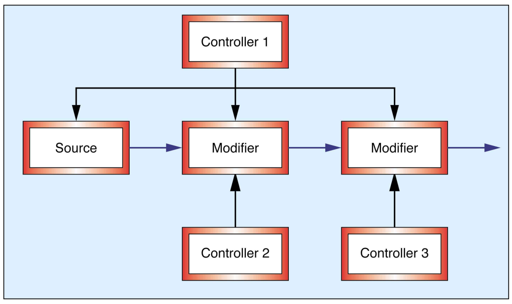


Figure 7: Viewing a simple synth as a set of sources, modifiers and controllers.

there are obvious examples of these, including contour generators, LFOs, pitch-bend wheels, and joysticks... all of which can transmit control signals to modifiers such as amplifiers and filters, as well as to sources such as oscillators.

We can therefore reconsider the synth architecture in Figure 2 and draw Figure 7 (above), in which each of the constituent parts is viewed in a much more general sense as either a source, a modifier or a controller. I like this representation because, although it is explicit about the shape of the patch, it does not fix the exact natures of the building blocks within it.

## Source, Modifier Or Controller?

Up to this point, I've kept everything simple, and although I've introduced the concepts of sources, modifiers and controllers, I haven't stepped far beyond the ideas of oscillators, filters, amplifier, envelopes and LFOs. But now I want to make everything more interesting by demonstrating that things are not always what they seem, and that there are many synthesizer modules that do not fit snugly inside a single classification.

Take, for example, the simple architectures shown in Figures 8 and 9 (below), both of which contain two audio frequency oscillators and a mixer. In Figure 8, the outputs from the two oscillators pass as audio to the mixer, so we can say that both oscillators are acting as sources. But in Figure 9, the output from the second oscillator is being fed into the pitch control input of the first, so we can say that while Osc1 is still a source, Osc2 is now acting as a controller. This means that the classification of the oscillator is not determined by its operation, but by its position in the patch!

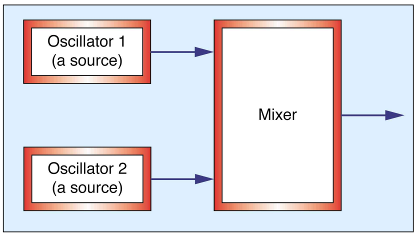


Figure 8: Two oscillators operating as sources.

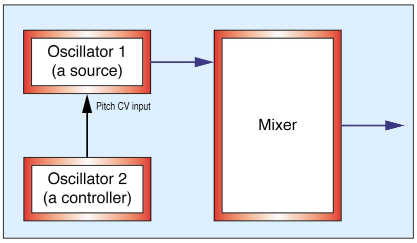


Figure 9: Two oscillators; one a source, the other a controller.

You don't need a huge modular synth to encounter this. A perfect example exists in the world's first pre-patched, integrated synthesizer, the Minimoog, on which a knob in the Controllers section of its front panel determines whether Osc3 is acting as a modulator (a controller) in addition to, or instead of, its role as an audio oscillator (a source).

Here's another example. Anyone who has read this series will be conversant with the fact that filters with two poles or more (ie. with 12dB-per-octave or steeper cutoff slopes) can exhibit 'resonance' and that, if pushed to the limit, the accentuation of the gain at the cutoff frequency will turn into sine-wave oscillation. This idea is illustrated very clearly in Figure 10 (above), which shows how you can use the oscillating filter to obtain a third pitch from a synthesizer — the Roland Juno 60 — that has just one oscillator. In other words, the filter is no longer acting just as a modifier, it is now a source.

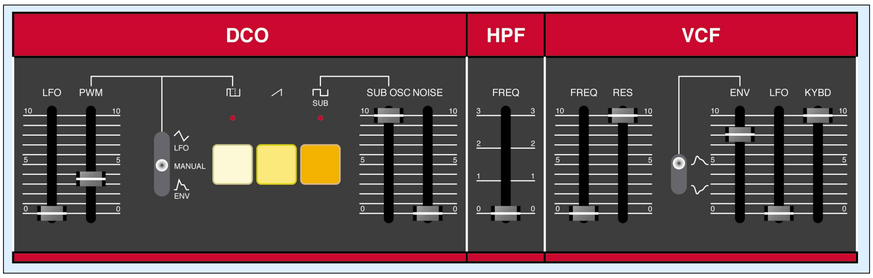


Figure 10: Using the Juno 60 filter as a source in an organ patch.

Clearly, you don't need a modular synth to think in a modular fashion, but thinking in these terms becomes even more useful as your synthesizer become more sophisticated. Take, for example, the trapezoid generator in the EMS VCS3. In normal operation, this acts as a contour generator; that is, as a controller. However, there is also a repeat mode in which the contour becomes a low-frequency cyclic waveform and acts as an LFO, which is another form of controller. But if you make the contour brief enough, the oscillation starts to move into the audio spectrum, at which point you could use it as either a controller *or* a source. It's now up to you to decide which is most appropriate, and to patch it into your sound in the way you feel to be most suitable.

So here we have three very different synthesizers, the groundbreaking Minimoog, the simple Juno 60, and the complex EMS VCS3, all of which contain one or more modules that can act ambiguously as sources, modifiers *or* controllers. The same also applies in the modulation matrices of modern digital workstations; oscillators become controllers, modifiers become sources... and so on. As you can see, 'modular synthesis' can be a valid idea on almost any type of synthesizer.

## Controlling Controllers

Now, there's nothing stopping you from modifying control signals, just as you modify the audio signals generated by the sources. Indeed, placing controllable amplifiers in the signal paths between controllers and the things they are controlling is the basis of almost all synthesis, and you will find many VCAs (or their digital equivalents) in even a modest synthesizer. But there's no reason to limit yourself to using amplifiers... you can filter control signals to create new control signals, or shift their frequencies, or chop them up using S&H units, and apply a thousand other ideas.

A simple example to illustrate this idea of controlling the controllers lies in Figure 11 (on the next page), which shows the filter section and contour generator of an ARP Axxe. In this case, the filter is a modifier, the ADSR envelope generator is a controller, and the three additional faders in the centre of the diagram (marked Kybd CV-S&H-Pedal, LFO, and ADSR) control the gain of three amplifiers (which are themselves modifiers) that determine how much of three controllers are applied to the modifier's cutoff frequency.

Yikes! That's a heck of a sentence, so let's look at Figure 12 (also on the next page), which is the block diagram for this patch. This is much clearer, and if you learn to

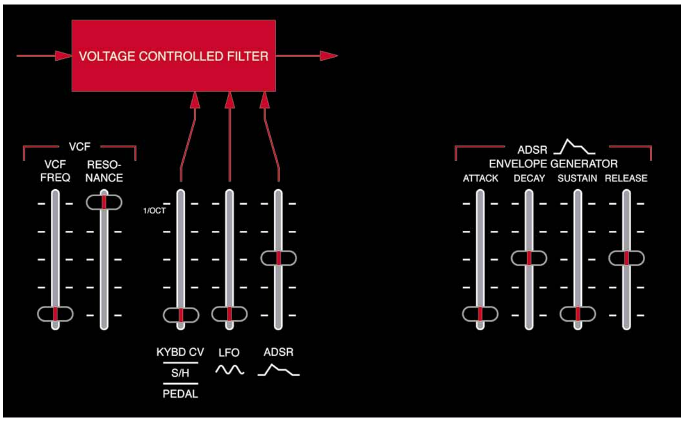


Figure 11: A modifier, some controllers, and lots of modifiers controlling the controllers.

think of your patches in this way, you can be very much more explicit about what is happening when you create a sound on a synthesizer.

## Strengths & Weaknesses

If you are going to apply modular thinking to a range of synths, you have to become aware of each one's limitations as well as its strengths. Let me give you an example...

'Sforzando' is the name of a type of sound in which the loudness peaks very quickly, dies away, and then swells later in the note. Generating this contour (shown in Figure 13, right) requires a minimum of five stages. Unfortunately, ADSR contour generators have only four stages. Perversely, each of the three envelopes on the cheap, often denigrated Korg EX800 has five stages, and many cheap digital synths have envelopes with five, six, or even eight stages, meaning that they are capable of producing a sforzando brass patch, whereas the 'mighty' Prophets, Jupiters, and Memorymoogs can not.

Consequently, you can't assume that all the facilities available on a cheap synth will be available on an expensive one, or that an expensive one can generate all the sounds that you can obtain from a cheap one. Nor can you make sweeping statements like 'analogue synths are better than digital ones'. If you want to create the sound of an overdriven low-pass filter sweeping through the spectrum of a sawtooth wave, it's probably true. If you want to imitate a brass instrument or an electric piano closely, or create a lush, evolving texture, it's probably not. You must therefore choose your instruments carefully so that you can produce those sounds if you want them.

## Why Bother?

Consider the following statement: the best way to develop a new recipe for a meal is to take random ingredients, throw them in a pot, and see if anything edible ensues.

It's rubbish, isn't it? If you were invited to a dinner prepared in this fashion, you wouldn't return for seconds. So, whether you want to imitate existing sounds or program interesting new ones, it's vital to be able to select the appropriate sources, the right sort of modifiers, and the controllers that can shape the sound and let you control it in the ways that you want. If you do not, it's likely that your random twiddlings will generate another filter sweep that sounds little different from a billion other filter sweeps created over the past four decades.

That's why I've spent the past three years showing you how it's possible to analyse sounds as diverse as brass instruments, guitars, orchestral percussion, electronic percussion, pianos, strings, woodwind and electro-mechanical organs, and how you might use the building blocks of common synths to emulate them. I've tried to make everything as general as possible, and have provided examples using a wide range of synths. Unsurprisingly, a handful of readers objected... they simply wanted me to show them how to set *this* voltage-controlled wotsit to *that* value, and thereby obtain the sound used on the latest chart-topping smash. In short, they wanted to know where the Big Red Button was. However, the point was not to show you the settings that synthesize a particular sound, but to teach you how to think in a modular fashion, no matter what synths you might own, and thereby allow you to work out the settings for yourself.

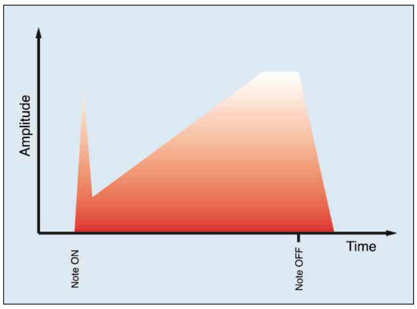
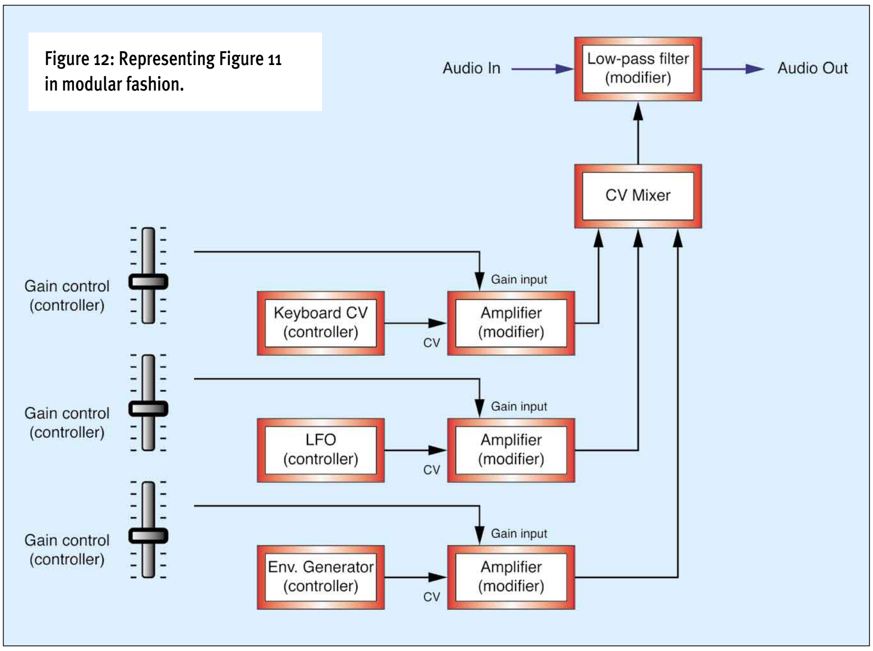


Figure 13: A five-stage sforzando envelope.

Of course, you may not be interested in understanding how and why synthesizers do what they do, and you may be perfectly content with serendipitous experimentation. If so, that's fine. But if you want to get the best from your instruments, I think you ought to go further than twiddling and hoping. Even modest synths allow you to think about synthesis in a modular fashion, so why not try it?

So — which *is* the best synthesizer? That's easy. It's the one that allows you to obtain the sounds you want. But whichever one you use, you should remember one last thing... there is *no* Big Red Button.

Figure 12: Representing Figure 11 in modular fashion.


Audio In
Low-pass filter (modifier)
Audio Out
CV Mixer
Gain control (controller)
Gain input
Keyboard CV (controller)
CV
Amplifier (modifier)
Gain control (controller)
Gain input
LFO (controller)
CV
Amplifier (modifier)
Gain control (controller)
Gain input
Env. Generator (controller)
CV
Amplifier (modifier)