Transoniq Hacker

The Independent Ensoniq Mirage User's Newsletter

FILTERS MADE FUN

By Clark Salisbury

Before we jump into this month's column, I think it would not be totally inappropriate for me to reminisce a tad - what with it being a new year and all.

The Mirage is now nearly a year old. The first Mirage that we got in at our store came complete with sound disk number one, period. No sustain pedals, no sequencer expander, nothing. Just a disk with piano, slap bass, flutes, a couple of drums, and some silly guitar sounds. In a little less than a year, though, the situation has changed a bit. Now there's MASOS, Input Sampling Filters, VES's (at least three - and growing), dozens of sound disks - and it looks like it's only the beginning for digital sampling. Lest we forget where we've been, does anyone remember the original Emulator? A cool $10,000 for a sampler with a four octave, non-touch-sensitive keyboard which would hold a maximum of two samples at any given time. It had no envelope generators, no dynamic filtering, almost no waveform manipulation, and of course no MIDI. Why, you couldn't even re-tune the samples by more than a few cents either way. And, still, it somehow managed to amaze us. Today there is talk of developing a part of the MIDI specification to allow for the dumping of sampled waveform data in a standard way so that you could take your Mirage samples and dump them into a Sequential Prophet 2000, or any other sampling device. This is particularly exciting news. Once the programmers and techno-weenies get hold of this, watch out! I don't think I exaggerate when I say that we are witnessing a revolution in the way sounds and music are being created and performed.

Anyway, this month I'd like to talk a bit more about filters and envelope generators and such, those friendly processors that help make sampling such good fun.

In Issue #4 of the Hacker, I briefly touched on the function of the Mirage filters, but, as many of you may have realized, there's quite a bit more to it than I could squeeze into that particular article. To briefly recap, the filters in the Mirage are 24 dB-per-octave resonant low-pass filters. What a mouthful. What this actually means is that the Mirage filters will attenuate (turn down) frequencies above their cutoff point at, say, the frequencies at 20 kHz (one octave higher) will be some 24db quieter. This, of course, is useful for eliminating unwanted noise and aliasing present in the original sample. Also, the filters are known as resonant filters because the frequencies at the filter cutoff point can actually be amplified. This can be used to emphasize upper harmonics present in the waveform, and is often perceived as quackiness, or the all too familiar "wah-wah" sound that has come to be identified particularly with analog synthesizers.

There are other uses for the filters as well. The sound of many acoustic instruments has a tendency to become darker over time as the upper harmonics decay. Percussive instruments, such as guitar, piano, and marimba are good examples of this. A looped sound, however, remains at the same level of brightness indefinitely, and even if a more or less natural decay in volume is obtained through the use of envelope generators and voltage-controlled or digitally-controlled amplifiers, things can sound somewhat unnatural if the overall brightness of a sound remains constant. This is one place where dynamic filtering can be of great usefulness. By progressively filtering the sound across time, a gradual (or not so gradual) change in the sound from bright to dark can be affected. And it works both ways. The filter can be used to make a sound get brighter over time. Or you can use rather unnatural envelope generator settings to get bizarre, cyclic-sounding changes in brightness, as with upper and lower program number 4 from the strings/cellos sample. So how do I control all these nutty kinds of changes, you ask? Easy. Just check out Uncle Clark's step-by-step guide to "Filter Wizardry," coming right up. RC090317

Let's think of the filter as a "brightness attenuator." It can be used to filter out higher frequency components in the waveform, while letting the lower frequencies pass through - hence the name "low-pass" filter. Note, however, that the filter cannot add upper harmonics or brightness to a waveform that does not already contain them; it can only work to attenuate what's already there. In theory, then, it is probably best to pre-emphasize the upper frequencies in any sounds you may wish to sample. Not to worry if your sample turns out a bit on the bright side - it can always be filtered down to normal brightness later on. And this type of filtering can have the extra benefit of filtering out unwanted noise and hiss from the wavesample. The point in the frequency spectrum at which the filter begins attenuating is called the cutoff point, and it can be controlled in a number of ways in the Mirage.

The most obvious filter cutoff point controller is the Manual Filter Cutoff control, Parameter 36. Increasing and decreasing the value of this control has the effect of raising and lowering the filter

cutoff point in semitone increments, and will be perceived as brightening and darkening the sound. (I know I said that the filter could not add brightness, but we first started with a sample that was overly bright, didn't we? If you want to know what I mean by overly bright, load the piano sample from the trusty old Disk 1 and set Parameter 36 to a value of 99 or so. See what I mean?) The next, and less obvious, filter controller to be aware of is Wavesample Relative Filter Cutoff (70). This is similar to the Manual Filter Cutoff control, but it affects the filter cutoff point only for the wavesample selected using Parameter 26, Wavesample Select. This control's raison d'etre is to give you a way to balance the brightness of multiple wavesamples by being able to set filter cutoff points independently for each of them. Nifty, huh? The third, and last, filter controller to worry about when setting the base cutoff point of the filter is Keyboard Tracking, Parameter 38. The idea here is that if you were to set a filter cutoff point that sounded appropriate on the low notes of the keyboard, you may experience problems with the high notes not sounding bright enough, because the higher harmonics present in the upper notes would be overly attenuated by the filter. Keyboard tracking can rectify this problem by actually using the keyboard to control the cutoff point of the filter. In other words, the higher you play on the keyboard, the higher the keyboard tracking will set the cutoff point of the filter. Parameter 38 controls the amount of this effect. To check this one out, simply load a sound into the Mirage and vary the value of Parameter 38 while listening to notes played first on the extreme lower end of the keyboard, then on the extreme upper end. RC090317

So far all the controls that we've talked about are used to set the base cutoff frequency, the point at which the filter cutoff is set when there is no modulation input. (I know, I know - keyboard tracking is a type of modulation. But for my purposes here, it's easier to group it with this set of controllers.) There are a number of other ways to control the filter cutoff point, as you may already have guessed, you sly devils, you.

The first of these is the envelope generator. We talked about these a couple of months ago, but I'd like to go into a little more depth here. There are 5 parameters to deal with in the Mirage envelope generator (not counting the velocity sensing parameters). They are Attack (40), Peak (41), Decay (42), Sustain (43), and Release (44). Attack (40) controls how long it takes for the filter cutoff to reach its highest level, with Peak (41) determining just how high the filter cutoff will go. In other words, if you want a sound to go from dark to bright at a fairly slow rate, you would set the attack parameter to some fairly high value. If you wanted the sound to eventually end up being pretty bright, you would set the peak parameter fairly high. Attack controls how long it takes to raise the filter cutoff point (get brighter), while peak determines how high (bright) the filter cutoff will actually end up going. We must remember, however, that the filter can not add brightness to a sound, only attenuate it. So if you have some other controller, such as Manual Cutoff (36) or Relative Filter Frequency (70) set to maximum, or if the cumulative effect of two or more controllers pushes the filter cutoff up to the maximum, any other controllers you may be using can have no audible effect. The filter simply has nowhere to go.

Next is the Decay Control (42), and it works in conjunction with the Sustain Control (43). The decay control determines how long it will take for the filter to go from the maximum value (set by the peak control) down to the value set by the sustain control. Thus, if the sustain control is set to 0, (and no other controllers are affecting the filter), the filter cutoff point will eventually reach a value of 0, (even if you continue to hold down the Mirage keys), effectively filtering out all frequencies. The effect is that the sound will continue to darken until it darkens itself right out of existence. If, on the other hand, you do not want the filter cutoff point to end up at 0, simply set the sustain control to a value higher than 0. This will have the effect of holding the filter "open" while the keys are being held, and the sound will sustain at this level of brightness until they are released. Last is the release control, and it affects the length of time it takes for the filter to finally decay to 0 once the key or keys have been released. Note that all the filter controllers discussed here, with the exception of the release control, are active only while the keys are depressed. As soon as you release the keys, the envelope generator immediately goes into the release position of its cycle, and all other control input to the filter is ignored.

But wait, there's more! Each component of the filter envelope has a velocity sensing (VS) counterpart. Attack VS (45) can be used to increase or decrease attack time according to how hard (alright - how fast) you play the keyboard. The effect is that if you play a key slowly, you can obtain a longer attack time than if the key is struck more quickly, allowing you to control, by touch, the length of time it takes for notes to swell. Peak VS (46) causes the filter cutoff point to go higher the harder the keys are struck, giving you touch control over brightness. Decay Kyb (47) is a bit different. In many acoustic instruments the decay time for high notes is shorter than for low notes . Decay Kyb causes the decay time to be affected by keyboard position; higher notes decay more quickly than low ones. The intensity of this effect is controlled by Parameter 48, Sustain VS. The harder you hit keys, the higher the filter cutoff point will be set, and your sound will sustain at a brighter timbre. Finally, we have Release VS (49). This one's kind of fun. With Release VS, the Mirage actually pays attention to how quickly you release the keys, and sets the release time accordingly. If you let go quickly, you get a short release, and the note decays quickly. If you let go more slowly, the note will take longer to decay. This can be a great effect on string samples.

It should be emphasized that the effect of all the filter controllers is cumulative; they are added together to form the actual filter cutoff point. With the exception of release and release VS, these controllers are active while one or more keys are

being depressed. Manual Filter Cutoff, Wavesample Filter Cutoff, and Keyboard are added together to form the base filter cutoff, (you can think of it as the basic brightness setting for your sample), with the effects from the envelope generators being added to this. And remember, if it doesn't make sense at first, mess with it till it does. Thanks for tuning in.

Clark Salisbury is Product Specialist with Portland Music Co. in Oregon, and is also a partner in "The Midi Connection," a Portland-based consulting firm. He has been actively involved in the composition, performing, and recording of electronic music for over five years, and is currently involved in producing and marketing his own pop-oriented compositions.

MAGIC SAMPLE RATES

RC090317

By J. William Mauchly

It is possible to get very clean high frequencies out of the Mirage. The usual technique is to oversample a sound; this is the best general solution to the problem of high frequency distortion. There is another technique, however, which only works for sounds that will not be transposed. This article explains how to get rid of all distortion by playing back a sound at the exact rate that it was sampled. It's great for really clean cymbals and special effects.

The Mirage uses a proprietary oscillator chip to create digital waveforms. The output of the "Q" chip is fed to eight analog filters, which reduce the high-frequency aliasing noise. Another kind of noise, however, is also present, which often cannot be filtered out. This distortion gets much worse as the high frequency content of a sound increases. The trouble is that the noise shows up anywhere in the spectrum.

If you are curious about this problem, it is instructive to try to sample a high sine wave. Try a pure sound about two or three octaves above middle C, and make sure the sample time is set for 34 (microseconds). When played back, you'll hear the aliasing. There is another sine wave present, probably lower than the original. It bounces around from note to note, but it does follow a strict rule. As you transpose the sound closer and closer to the pitch of the original sound, the alias gets lower and lower in pitch. That's interesting - can you make it go away? Find the note on the keyboard where the alias tone is the lowest pitch. By using the fine tune control [68] of the wavesample, or master tune [21], you can get the alias to go subaudio. Alas, this is still not low enough; now the super-low frequency creates a tremolo effect as it interacts with the real signal. Here is the answer: when the alias goes to 0 Hz it will disappear.

Ah, but it's not quite that simple. The tuning of the Mirage is calibrated in 256ths of an octave. There is no "notch" in the tuning software which is exactly right to do the job. You get close, but not close enough; the distortion is still there.

Now the trick: the detune parameter [33]. This parameter detunes oscillator two in the smallest increments available in the hardware. These are smaller steps than those in the wavesample fine tune parameter. Set the mix [34] to 63, so that only oscillator two is heard. Now bumping the detune parameter can get you to that magic frequency where the distortion disappears. Completely. What's happening is that you are setting the playback sample rate of the sound so that it is EXACTLY the sample rate of the Q chip, which is 1/34 microseconds.

Now I warned you that that sine wave only sounds pure on that one key. But one key is enough for certain sampled sounds; a ride cymbal is a perfect example; a bell-tree is another.

Here are the constraints, in brief:

1. The sound must be sampled with the sample time at 34.
2. The mix [34] must be at 63.
3. The detune [33] should be set at 1. (Other combinations may work.)
4. The fine tune must be set according to this chart, depending on what key you want the sound to come back on:

KEY	Fine Tune [68]
C	4E
A	8E
D#	CE
F#	0E

You can find the right value of other notes by experimenting. The octave should also be set by ear. Octave transpositions will also be cleaner and have less distortion.

Admittedly, it's an awfully obscure technique. But it really does work; it sends the alias down to 0 Hz on that one key. The other technique, oversampling, reduces the amplitude of the alias. That is obviously a better answer for most sounds. I hope this tidbit will be of some use to those hackers who are always searching for the ultimate highs.

Bio: J. William Mauchly, son of the co-inventor of the ENIAC (the first digital computer) Dr. John W. Mauchly, has a degree in Computer Science from Temple University and is Senior Software Engineer at ENSONIQ Corp. After playing guitar and synthesizer professionally for several years, he became interested in computer music and digital processing. He started doing Fairlight consulting and microcomputer music programing around 1980. Since then, he has been music director for the Symposium on Small Computers in the Arts, and has been with ENSONIQ for two years. He was one of the designers of the Mirage and the ESQ-1, and author of the Mirage Advanced Samplers Guide.

How I learned to Love Hexadecimal

By Jim DuLaney

Since the procedures for creating multisamples with MASOS are so dependent on a little knowledge about hex math, a little information on that subject might be useful. As a professional in the computer industry, I first had to come to grips with this about 15 years ago. I know that during the 70's there was a brief flurry of activity in "new math" or math based on numbering systems in bases other than 10. To those of you who failed this, or never had it, this article is dedicated.

Our familiar numbering system is a base 10 numbering system. In a base 10 numbering system, there are 10 single digit numbering representations (0-9), and there is no single digit which represents the number"10". The number "10" is the first number which causes an overflow to more than one digit. Simply stated, the mechanics of the base 10 numbering system are as follows:

```
Thousands  Hundreds  Tens  Units
   X          X       X     X
   :          :       :     This number * 10 exp 0,
   :          :       :     (1), plus
   :          :       :
   :          :       This number * 10 exp 1, (10)
   :          :       plus
   :          :
   :          This number * 10 exp 2, (100)
   :          plus
   :
   This number * 10 exp 3, (1000)
```

Equals the number represented by the Xs above.

In a base 16 numbering system, which is what hex is, there are 16 single digit numbering representations (0123456789ABCDEF), and there is no single digit which represents 16 which is the first increment where numeric overflow to two digits occurs. The mechanics of base 16 are the same as base 10 representation:

```
x4096   x256     x16     units
  X       X       X       X
  :       :       :       This number * 16 exp 0 (1)
  :       :       :       plus
  :       :       :
  :       :       This number * 16 exp 1 (16)
  :       :       plus
  :       :
  :       This number * 16 exp 2 (256)
  :       plus
  :
  This number * 16 exp 3 (4096)
```

Equals the number represented by the Xs above.

Each X could be a value from 0 - 15 (0123456789ABCDEF)

The obvious question at this point becomes "why bother with hex?" The answer is not quite so obvious. What I will try to show you here is that binary and hex are the same thing.

Among computer techno-weenies hex is an easier way of handling binary (base 2) which is what the computer really understands. A bit is the smallest significant unit of data storage in a computer. It is an entity which, like a coin flipped, can only be heads or tails. A bit can store a value of zero or one, and nothing else. Bytes are a collection of bits of a specified number (depending on the machine in question). In the Mirage, a byte consists of eight bits, and is entered, displayed, and manipulated in hex. Here's how it works. RC090317

In base two, each individual digit can only contain one of two values (0 - 1, off- on). In binary math (base 2), there is no single digit which represents the number two and the number 2 is the first time overflow dictates a two digit representation.

```
8      4      2      1
X      X      X      X
:      :      :      This number * 2 exp 0, (0)
:      :      :      plus
:      :      :
:      :      This number * 2 exp 1, (2)
:      :      plus
:      :
:      This number * 2 exp 2, (4)
:      plus
:
This number * 2 exp 3, (8)
```

equals the number represented by the Xs above. Each X could be a value from 0 to 1

From this diagram you can see that the highest decimal number which can be stored in a four bit "nybble" (half a byte) is 8 + 4 + 2 + 1 = 15. An eight bit byte, broken into two nybbles neatly displays as a two digit hex number. Pretty nifty, huh? Once you understand these basic mechanics it all becomes pretty clear.

Decimal	Hex	Binary
1	01	0001
2	02	0010
3	03	0011
4	04	0100
5	05	0101
6	06	0110
7	07	0111
8	08	1000
9	09	1001
10	0A	1010
11	0B	1011
12	0C	1100
13	0D	1101
14	0E	1110
15	0F	1111