Lynx Sound Overview: This document is an overview for those who have had trouble seeing the forest for the trees. You should also read page 29 of the handy hardware spec. and the handy audio hardware overview starting on page 35 of the Handy software programmer's guide and notes. And of course handy appendix 2 hardware addresses FD20 through FD50.

1. What in the world were we thinking when we designed the sound hardware?

The original design goals were that the circuits:
1. be cheap
2. require relatively little cpu help to make useful game noises
3. have sufficient range and accuracy for tolerable music
4. have 4 channels
5. have direct access to the DACs.
6. be cheap.

Our approach was similar to the Atari 800 or 2600 sound system.

First we gave it 4 polynomial counters. What are they?

The Lynx system clock is 16 Megahertz, a bit high for music on this planet. So we pass the clock through some dividers to knock it down. First are some binary prescalers, then into the dreaded poly counter. The poly counter is really just a programmable divider. A binary counter (the normal kind) would divide by 7 by counting 0,1,2,3,4,5,6,0,1,2,3,4,5,6 etc. A polynomial counter might count 0,2,5,4,3,1,6,0,2,5,4,3,1,6 etc. It has the same number of states but they come in a pseudo-random order. The repeat period is programmed by selecting the initial value in the shift register (set shifter) and by picking which feedback taps are connected.

Why poly? Don't you like binary?

Binary is fine if you like square waves. Suppose you want a motor sound for a race game. A square wave tone is clearly not sufficient. A poly with a fairly quick repeat period will introduce a slight garble to the pure tone making a better engine sound. Once started the sound continues by itself with no additional CPU work. To increase the motor's RPM, keep the same shifter set-up and just change the frequency it's clocked with (the prescaler). More on this below. Thus with minimum CPU work we can make a complex engine sound that changes RPM smoothly.

Next we gave the CPU direct access to the four 8 bit DACs. A running poly counter uses a DAC for its output, but we wanted to give the programmer the flexibility to write to unused channels directly. There has been some confusion about the DACs since they are used with the poly system to sort of make 127 loudness levels. The truth is they can be used as 4 independent true 8 bit DACs.

Finally there is a one pole RC filter in hardware on the output at about 4500 Hz. Future stereo versions anticipate a bass boost circuit.

2. What in the world were we thinking when we designed the sound software?

We were thinking let's use existing Epyx sound tools wherever possible. SPL (Sound Programming Language) was designed by music people for music people, they liked it so we adopted it. Of course music people think music is the most important thing, and they insisted a 240 Hz. update rate was the minimum acceptable for sounds. Many programmers have gawked in disbelief at the size and cpu usage of the sound driver. Well, it all depends on your point of view. Programmers are of course free to write their own drivers to meet their own needs.

3. That's great, but I still don't have a clue what sounds will come out of HSFX.

Ok Ok, here's some more detail. First you should bring up the HSFX Edit screen.

There are three main areas I'll mention. The box in the upper left is the clock source for the poly counter. The poly counter is in the lower left box. To the right of the poly counter is a box that says what to do with the poly's output.

Key concept #1: Wherever you see the word "interpolate" that function is done in software by the sound driver. It is not a hardware function. 240 times a second the sound driver does a linear interpolation slewing from the first value to the next. You'll also see the word "Integrate". This is done in hardware, and I'll try to explain later.

Key concept #2: "Frequency" isn't the frequency of the note, but rather the frequency used to clock the poly counter. For a given poly setting lowering the frequency value will lower the frequency of the output sound. (good for engine RPM changes)

So, starting with the first box, the various options let you select a clocking rate in a straightforward binary manner. I don't really think anyone has a problem with this, if you do, you might as well stop reading since you won't have a prayer of understanding the ----

POLYNOMIAL COUNTER!!!

The shift register (shifter) with taps into an EXOR feeding back into the shifter together make up the polynomial counter. There is little intuitive about its operation, except that the farther to the left you select taps, the more random the sound can be. (not will be - can be)

It may not be too helpful but I've attached 2 printouts showing all the possible combinations of the poly counter. Ideally this data would be integrated in a transparent way into HSFX - but don't hold your breath.

The first printout lists the combinations in order of shifter period, that is the number of counts until the counter loops back to its initial value. Notice this number ranges from 1 to 4095. Observe that there are 19 different ways to get a period of 4095. All 19 ways have an initial shifter value of 00 and a different set of feedback taps. All will repeat at the same rate, but will sound slightly different (it's actually hard to hear the differences

in this example since 4095 is the longest and therefore most random count). Loop counts of 1 are lockup states, and not too interesting. In general the closer the loop count is to 2 the more the tone will sound like a square wave. The closer the count is to 4095 the more the tone will sound like white noise.

The second printout lists the combinations in feedback tap order. So for example a feedback tap setting of $02E can count in any one of 8 different sequences depending on the shifter initial value. Initial values of $000, $005, $00B all have a period of 15, but will sound different (let's say they have different timbres). Values of $002, $00E, $016 have period 5 and different timbres. $009 has period 3, and $03F is a lockup- period of 1.

The output of the poly counter is just a single bit. It is used in one of two ways, depending on whether integrate mode is set or not.

In normal nonintegrate mode, the bit selects either the value in the volume register or its 2's complement and sends it to the output DAC. For example, suppose the poly is set to make simple square waves, and the volume register has a 9 in it. The bit stream out of the poly counter will be 111000111000 etc. and the output DAC will see 9,9,9,-9,-9,-9, 9,9,9,-9,-9,-9 etc. Increasing the value in the volume register will make louder square waves up to a max of 127,-127.

In integrate mode, instead of sending the volume register directly to the DAC it instead adds the volume register (or it's 2's complement) to a running total that is then sent to the DAC. So, assuming we start at 0, our example would be 111000111000 yields 9,18,27,18,9,0,9,18,27,18,9,0. Notice that this makes a more triangular wave. This sounds different from a square wave. The running total clips rather than wrap around (which would sound horrible).

The 3 interpolation choices in HSFX can now be understood. Interpolation of frequency will cause a glide from one note to the next. Interpolation of volume makes smooth transitions between loudness. And interpolation of feedback taps makes no sense at all. However it does make an interesting breaking glass sound characteristic of the LYNX. Hey, it was easy to throw it in the driver.

4. Enough already, I'll write my own driver. I want to talk directly to Mikey.

Fine by me let's just take a quick look at the hardware.

In Mikey all the timers and sound channels share one piece of hardware. Each timer/ sound channel takes its turn rotating through the hardware. This is why it sometimes takes as long as 1 usec to access these registers, you have to wait for it to take its turn. This is also why the sound and timer registers behave so similarly.

The prescaler registers FD24 and FD26 behave exactly as any timer. FD25 is almost the same as a timer The only differences being bit 7 where we stuck one of the feedback taps (no interrupts on sound) and bit 5 which controls integrate mode. The lower nibble of FD27 works the same as a timer, except for bit 3 whose function makes no sense for audio. It won't matter to you, since the only time you'll write to FD27 is when you are

initializing the shifter, and the sound channel has to be turned off then or you won't be able to set both registers of the shifter before they change.

Speaking of the shifter it sits at FD23 and FD27. To repeat, make sure the poly counter isn't clocking before trying to initialize these registers.
The feedback taps are set at FD21 and bit 7 of FD25.

Bit 5 in FD25 controls integrate mode. FD20 is the volume control. Remember in integrate mode the volume is added to a running total and sent to the DAC. The running total is kept in FD22. In normal mode FD22 will contain volume or its 2's complement. Notice that you don't normally need to play with FD22, it just gets what it needs. If you want to store directly to the DAC then FD22 is the place to do it. You'll probably want to shut off the audio timer before storing to FD22.

5. Stereo, stereo where for art thou stereo.

Originally Lynx was mono. After Mikey was working and ready to produce, it was decided to add simple stereo. The Howard boards were not yet finished, so we went ahead and implemented this stereo on them. This form of stereo was channel switching controlled by FD50. Later it was decided to add panning and attenuation registers FD40 through FD44. Attached is a sheet detailing these registers to be added to your appendix 2 hardware addresses. A plug in upgrade board for Howard boards was designed and is available from Atari. As of today 4/25/91 no stereo lynx have been produced. In fact the potential existence of stereo is still confidential. Klax supports full stereo panning, and digitized sound. Xenophobe supports some amount of stereo, and maybe some other games do too - who can remember?

6. As usual Craig, you've taken a difficult subject and made it elegantly simple.

True, true. For the answers to more sound hardware questions you can leave messages for me on the Atari Sunnyvale bbs, or call me at EPYX 415-368-3200 FAX 415-369-2999.
Craig Nelson @ EPYX.