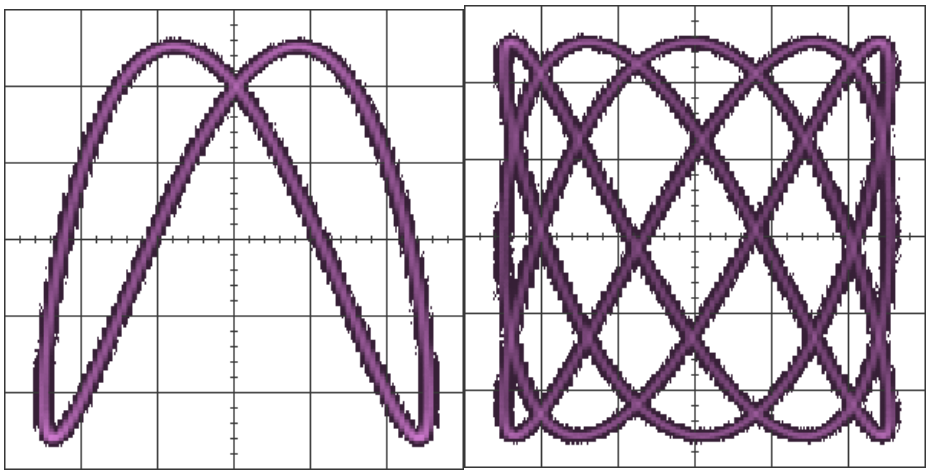
GENERATING LISSAJOUS CURVES

It is a truth universally acknowledged, that a person in possession of an oscilloscope must draw Lissajous curves.



As article explains a Lissajous curve is a parametric plot of sinusoids:

$$x\ y=\sin(a\omega t+\phi),=\sin(b\omega t).$$

To draw nice curves we want a/b to be rational, and for the whole thing to be precise and stable.

In the past, one might have used use fancy analogue electronics to generate these sinusoids, but today there are good digital alternatives.

In particular, we will:

- generate the signals with a couple of AD9850 DDS synthesizers;
- control these with an Arduino;
- buy everything prebuilt from eBay because it's easier and probably cheaper than getting the parts separately.

With all this we can set the sinusoids' frequencies anywhere up to about 20MHz with a precision of about 0.03Hz.

Software

There are many AD9850 drivers on the Internet. It falls into that class of hardware which presents just enough complexity that it's helpful to start from working code, but is easy enough that there's nothing much to encapsulate.

Many examples use a static singleton AD9850 device, which makes it harder for our application. However, Poul-Henning Kamp wrote something more suitable which I proceeded to clone and butcher. You can see the result on GitHub.

There are three key changes:

1. I added support for a reset line;
2. I extended the API to facilitate generating sinusoids with frequencies in a rational ratio;

3. I added an example to drive a couple of synthesizers over a serial connection to the Arduino.

On top of this I changed the API's style to suit my own preferences.

The key API change is to explicitly expose the integer which sets the AD9850's frequency. Thus we can easily ensure that e.g the synthesized frequencies are in the ratio 3:2 without worrying about how they'll be rounded.

For example, to set the oscillators to 200kHz and 300kHz we might do this:

```
AD9850 osc1(...);
AD9850 osc2(...);
double f = 100000.0;
uint32_t base = osc_1.calc_phase_delta(f);
osc1.set_phase_delta(2 * base);
osc2.set_phase_delta(3 * base);
```

If the 125MHz master clock were exactly correct we would see output frequencies about 0.0095Hz and 0.0142Hz too high. However regardless of the master clock their ratio will be exactly 2:3.

Source: http://www.mjoldfield.com/atelier/2015/06/ad9850-lissajous.html