Digis R Eazy

- Written by Decomp/Style.
- Taken from the C64 mag Domination #13
- Converted to text by Jazzcat/Onslaught.)

Digitized sound was one of the most impressive things you could do with your 64 back in the golden age of the scene. But they're complicated and too hard for any but the elite to understand, right?

No way!

A digitized sound player in its basic form is one of the simplest routines there is. There are lots of concerns and concepts to master in order to produce high quality sound, but the basics are simple enough for anyone.

We'll begin with a description of how digitized sound is played and then present a routine that does the job.

To start, a digitized sound file is just a collection of samples. Every so often, the amplitude of the soundwaves is measured and stored. To play the sound, we just write that sample to the volume register on the SID chip and delay a bit until it's time to write out the next sample. That's it. All we have to do is get the delay close enough for it to sound right. Even if the delay value is off a bit, you can still recognize the sound, it's just transposed to a higher or lower pitch.

We need to understand the file format of a digi sample in order to play it correctly, as well as have a bit of knowledge about the SID.

The SID's volume register is located at memory $D418. The upper four bits control the filters but can be set to all zeros for our purposes.

The .RAW file format is designed around that four bit volume register. It consists only of 4 bit samples, with two samples stored in each byte. The first sample is stored in the upper four bits. To make our first routine easier, we can play only the samples stored in the low nybbles. We lose some crispness in our output, but the sound is still acceptable.

```
start:  lda #<sample
    sta $FE
    lda #<sample
    sta $FF
    ldy #0
-   lda ($FE),y
    sta $D418    ; play sample nybble
    ldx #delay    ; controls pitch/speed
-   dex
    bne -
    iny
    bne --
    inc $FF
    bne --
    rts
```

Now, we just load the sample into memory and run the routine. If the sound isn't right, adjust the delay value and try again. If the sample sounds too slow and deep, lower the delay value.

This routine is far from being perfect. Its major flaw is that it keeps playing past the end of the sample, producing random noise. But from this start, you can produce a reasonable routine to play back sampled sound effects and even music to a limited degree.

Decomp/Style

http://www.cei.net/~rreed/ (Editor note: This URL is broken since long time now.)