Making noise with a SN76489 Digital Sound Generator – Part 2

By marco_c < https://arduinoplusplus.wordpress.com/author/arduinoplusplus/>

October 19, 2019 < https://arduinoplusplus.wordpress.com/2019/10/19/making-noise-with-a-sn76489-digital-sound-generator-part-2/>

In the first part < https://arduinoplusplus.wordpress.com/2019/10/05/making-noise-with-a-sn76489-digital-sound-generator-part-1/> we examined the basics of the SN76489 hardware and how to manage it at the hardware interface between MCU and IC.

To enable sound generation experiments, the first thing I did was create a library to allow me to write sketches without worrying too much about this underlying hardware management.

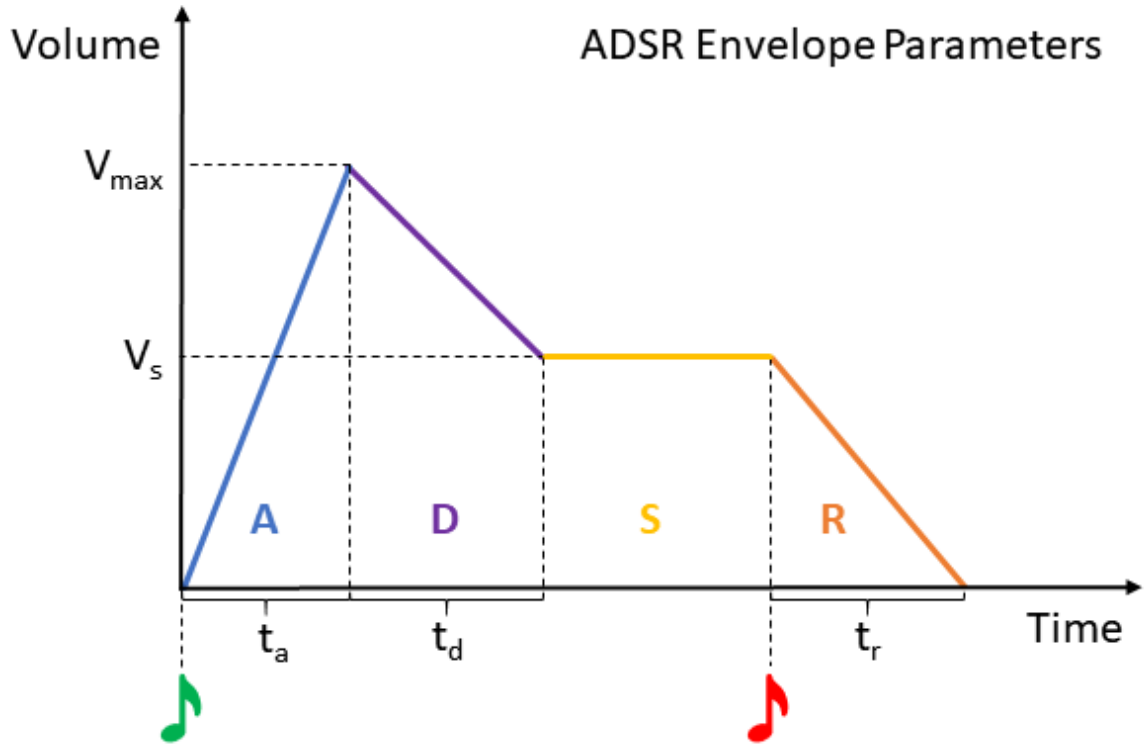
The MD_SN76489 library < https://github.com/MajicDesigns/MD_SN76489> and the example code I created during my experiments can be found in my code repository < https://github.com/MajicDesigns> . However, before we get into it, it is worth understanding some underlying concepts.

Some notes about Notes

The Attack, Decay, Sustain, Release (ADSR) Envelope

An ADSR generated sound envelope is a component of many synthesizers and other electronic musical instruments. The sound envelope modulates the sound, often its loudness, over time. Physical (blown, hit or plucked) instruments have the characteristic curve shown in the figure below, although the A, D and R components are more naturally exponential curves.

Electronic instruments, however, can also implement an inverted ADSR envelope, resulting in the opposite behavior: during the attack phase, the sound fades out from the maximum amplitude to zero, rises up to the value specified by the sustain parameter in the decay phase and continues to rise from the sustain amplitude back to maximum amplitude.



An ADSR envelope can be specified using the following parameters:

Attack	The time interval (Ta) between activation and full loudness (Vmax).
Decay	The time interval (Td) for Vmax to

	drop to the sustain level (Vs).
Sustain	The constant sound volume (Vs) for the note until it is released.
Release	The time interval (Tr) for the sound to fade from Vs to 0 when a note ends.

As the SN76489 only produces pure tones, ADSR modulation must be implemented in software. For flexibility, these ADSR envelopes are replaceable at run time on a per channel basis, and we can invert the specified ADSR curve.

The SN74689 volume controls are also limited to 15 steps, so the A, D or R phases are easiest implemented as linear interpolations of sound volume (equivalent to sound power output) over time.

Playing a Note

An ADSR note cycle is always controlled by 2 events – an *note on* event (the green note in the ADSR diagram above) that triggers the attack phase of the cycle and a *note off* event (red note above) that triggers the start of release phase of the cycle. On a

keyboard instrument these would correspond to a key being pressed and then released.

Following some research, I concluded that electronic music files seem to come in two varieties for how a score is encoded in the file:

- A **sequence of note on and note off** events (eg, MIDI files). In these files, the note remains in the Sustain phase until the note off event at which time the note is Released.
- A **note and a duration** (eg, RTTL tunes). In these files the note off event is implied by the duration, noting that it already includes the Release time and requires us to back-calculate the Sustain period.

So the library also needs to be flexible in how the note is played, as well as the ADSR parameters for the tone generator.

Writing some code

The **MD_SN76489 < https://github.com/MajicDesigns/MD_SN76489>** library is not overly complex. The majority of the code is a FSM to keep track of the ADSR envelope as a note is played.

The library allows both direct connection and SPI through a '595, although the majority of testing was carried out with the direct connect shield I described in the first part of this series. In both cases the object definition include a list of all the I/O pins that are used to connect the MCU to the SN76489 IC.

To keep the sketch code as generic as possible all the ADSR envelopes and/or automatic note off events are managed by the library. For this to happen in a timely manner, the *loop()* function must invoke the *play()* method every iteration through *loop()*. The method executes very quickly if the library has nothing to process, imposing minimal overheads on the user application.

Also, the library allows the generation of tones (pure frequency sound), notes (frequency and ADSR envelope) and noise (using the noise generator) using the *tone()*, *note()* or *noise()* methods respectively. These can be specified:

- **Without a duration parameter** to to indicate that the user code need will generate the note off event
- **With a duration parameter** to indicate that the library should generate a note off event at the end of the specified total duration. If an ADSR envelope is active, the note duration encompasses the time between initial Attack phase (note on event) to the end of the Release phase.

Additionally, the example sketches use a trivial helper class/library **MD_MusicTable < https://github.com/MajicDesigns/MD_MusicTable>** to look up musical note frequencies.

In the **next part < https://arduinoplusplus.wordpress.com/2019/11/03/making-noise-with-a-sn76489-digital-sound-generator-part-3/>** we'll examine what the output of all this work looks and sounds like.

Up ↑