Making noise with a SN76489 Digital Sound Generator – Part 1

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

October 5, 2019 < https://arduinoplusplus.wordpress.com/2019/10/05/making-noise-with-a-sn76489-digital-sound-generator-part-1/>

Most computer games from the 80's are recognizable by the bleeps and bloops they produced for sound. The easiest way to do this to toggle a single I/O pin to generate a square wave but there are some retro sound ICs that allow us to do much better for a minimal investment.

The SN76489 is one such IC that is still available at a very modest price and is easily interfaced to modern microprocessors.

The SN76489 IC

The SN76489 Digital Complex Sound Generator (DCSG) is a TTL-compatible programmable sound generator chip from Texas Instruments. It provides:

- 3 programmable square wave tone generators (122Hz to 125kHz)
- 1 noise generator (white noise and periodic noise at 3 different frequencies)

- 16 different volume levels
- Simultaneous sounds on one audio output

Its main historical application was the generation of music and sound effects in microprocessor systems. It was extensively used in early game consoles, arcade games and home computers.

The IC is a 16 pin DIP package with the following pinout:

Signal	Description
D0-D7	Command byte inputs
/WE	Active low Write Enable (latches data)
VCC	5V
GND	Ground
AUD	Audio output (headphone jack)
CLK	4MHz clock signal (see below)
/CE	Active low Chip Enable (connect to GND or MCU output if more than one IC shares D0-D7)
RDY	Ready signal (unused).

If multiple independent ICs are interfaced, then the ICs CE line is used to select the right device so that the data lines are shared.

Tone Generators

The frequency of the square waves produced by the tone generators on each channel is derived from two factors:

- The speed of the external clock.
- A value provided in a control register for that channel (called N).

With an external clock frequency of 4MHz, each channel's frequency is arrived at by dividing the external clock by 32 and then dividing the result by N. Thus the overall divider range is from 32 to 32768.

This gives a frequency range at maximum input clock rate of 122Hz to 125kHz (a range from roughly A2, two octaves below middle A, to 5-6 times the generally accepted limits of human audio perception).

The clock signal may be supplied from external hardware, or it can be created by the MCU with a timer capable of generating the 4MHz clock. This is generally possible using Arduino AVR hardware, where some assembly level programming enables a timer to toggle a digital output at the required frequency. Generating the clock with the processor makes this an one IC sound generation system.

The technical specifications for this IC have been pored over at length by retro computer enthusiasts. Aside from the usual **product datasheet < http://members.casema.nl/hhaydn/howel/parts/76489.htm>** , one of the most complete information references for this IC is the **SMS Power web site < http://www.smspower.org/Development/SN76489>** .

Test hardware

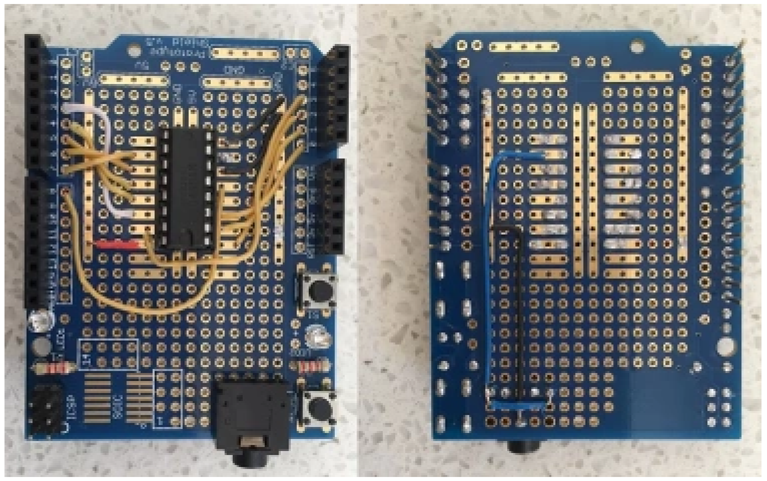
Directly connecting the IC to the MCU is a straightforward exercise in point-to-point connections between MCU outputs and IC inputs. This direct connection uses 8 digital output data lines from the Arduino MCU and an additional Write Enable digital output to load the data into the SN76489 IC.

I mapped the connections between the MCU and SN76489 as shown below. The numbers in square brackets are the Arduino or SN76489 pin identifiers. The Arduino pins are arbitrary except the 4Mhz clock.

Arduino Pin	SN76489
D0 [A0]	D0 [3]
D1 [A1]	D1 [2]
D2 [A2]	D2 [1]

Arduino Pin	SN76489
D3 [A3]	D3 [15]
D4 [4]	D4 [13]
D5 [5]	D5 [12]
D6 [6]	D6 [11]
D7 [7]	D7 [10]
WE [8]	/WE [5]
4Mhz [3 for Arduino Uno]	CLK [14]
GND	/OE [6]
	AUDIO [7]

The test hardware was built on an Arduino shield (shown below). In hindsight a different mapping could have avoided the yellow wires crossing over. We live and learn!



[< https://arduinoplusplus.files.wordpress.com/2019/10/sm76489_shield.jpg>](https://arduinoplusplus.files.wordpress.com/2019/10/sm76489_shield.jpg)

SPI Connection the the MCU

An alternative way to connect the MCU is using SPI through a 74595 Serial to Parallel buffer IC. This uses 3 SPI pins (LD, CLK and DAT), an additional Write Enable digital output to load the data from the '595 outputs to into the SN76489 IC and the 4Mhz clock pin if the MCU is generating the clock signal.

This uses less pins than a direct connection but requires an additional IC. This setup was tested in a breadboard and not built into a permanent circuit.

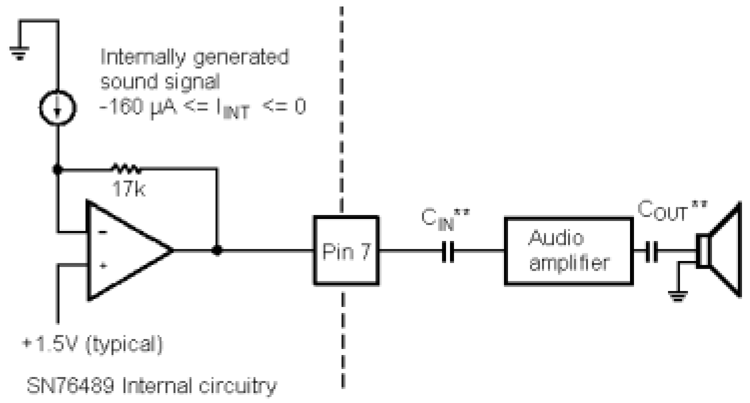
I mapped the connections between the MCU, 74595 and SN76489 as shown below. The numbers in square brackets are identifiers and, once again, the Arduino pins are arbitrary except the 4Mhz clock.

Arduino Pin	74595 buffer	SN76489
	D0 [15]	D0 [3]
	D1 [1]	D1 [2]
	D2 [2]	D2 [1]
	D3 [3]	D3 [15]
	D4 [4]	D4 [13]
	D5 [5]	D5 [12]
	D6 [6]	D6 [11]
	D7 [7]	D7 [10]
Data [11]	DAT [14]	
Load [10]	LD [12]	
Clock [13]	CLK [11]	
GND	/OE [13]	/CE [6]
+5V	/MR [10]	
WE [8]		/WE [5]
4Mhz [3 for Arduino Uno]		CLK [14]

Arduino Pin	74595 buffer	SN76489
		AUDIO [7]

Audio Output

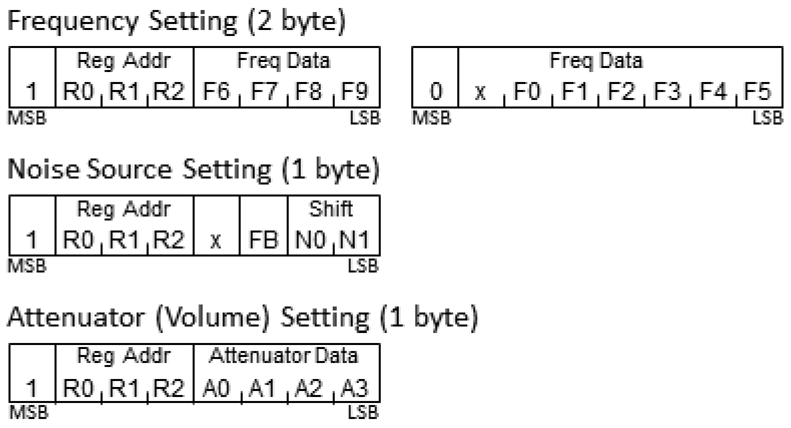
The Audio output from pin 7 of the IC is a mono signal that can be directly fed into earbuds (10mA max) or to an external speaker through an amplifier.



Communicating with the IC

The SN76489 has 8 internal registers used to control the 3 tone generators and the noise source. Data transfers are made one byte at a time in parallel through the 8 data bits D0-D7, using the control lines for handshaking.

For data transfers, the first byte has the MSB set and contains a three bit field which determines the destination control register and data for that register (shown below).



There are 3 types of data messages:

- To set the sound frequency, the tone generator requires 10 bits of information (the N described above). A frequency change requires a double byte transfer – the second byte does not have the MSB set.
- The noise channel does not use the same frequency field but instead allows selection of white or periodic noise (FB bit) and two bits for frequency control. This is a single byte transfer.
- Each channel's volume can be controlled using 4 bits of information to set sound attenuation for that channel. This is also a single byte transfer. Note that a higher attenuation implies a lower volume.

To transfer bytes between MCU and the IC

- /CE is set low to select the right IC.
- MCU output bits are set to the desired values for D0-D7
- /WE is strobed to load the content to the appropriate control register.

The SN76489AN requires 32 clock cycles (8ns at 4MHz) to load the data into the control register, which must be allowed for in the MCU interface software.

Now we've established how to work with this hardware, in the **next part < https://arduinoplusplus.wordpress.com/2019/10/19/making-noise-with-a-sn76489-digital-sound-generator-part-2/>** we'll start using this IC to make some sounds.

Up ↑