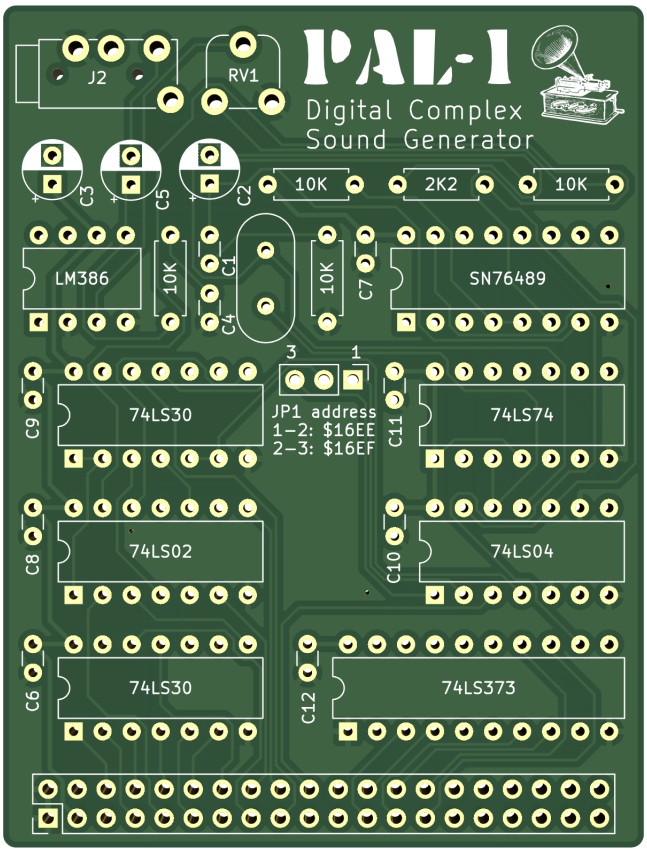


PAL-1 Digital Complex Sound Generator

Plastic Objects Limited

Published by **Plastic Objects Limited**
Woodbridge, UK

August 2022

Disclaimer

Contents

Introduction

The PAL-1 Digital Complex Sound Generator is an expansion module for the PAL-1 system[1], based on the Texas Instruments SN76489 Digital Complex Sound Generator (DCSG)[2] and was inspired by a Ciarcia's Circuit Cellar project[3].

The SN76489 DCSG was used in a range of arcade games, game consoles, and home and business computers from the late 1970s and throughout the 1980s. Notable systems using discrete SN76489 ICs include the ColecoVision games console and BBC Micro home computer. Others, such as the SEGA Mega Drive, include the SN76489 in their Video Display Processor (VDP) or used SN76489 clones, for example the Tandy 1000 and IBM PCjr.

The SN76489 provides three programmable tone generators and a noise source. The signals generated by the tone generators and noise source are mixed in the IC and passed to an output buffer. Since the maximum drive of the SN76489 output pin is just 10mA, the PAL-1 Digital Complex Sound Generator includes a simple audio power amplifier circuit to allow a speaker to be directly connected.

[1] Liu Ganning, 'PAL-1 Microcomputer User Manual' (November 2020), ⟨`http://pal.aibs.ws/assets/PAL_en.pdf`⟩ accessed 2022-07-21.

[2] The Engineering Staff of TEXAS INSTRUMENTS Semiconductor Group, 'SN76489AN' (TEXAS INSTRUMENTS, N.D.).

[3] Steve Ciarcia, 'Add Programmable Sound Effects to Your Computer', *Byte*, 7:7 (July 1982), 60-72.

1. *Getting Started*

Board assembly

Before assembling your PAL-1 Digital Complex Sound Generator board check the package contents against the Bill of Materials on page 19, and contact your distributor as soon as possible if any items are missing.

No specialist tools are required for assembling the board, though care must be taken when handling ESD sensitive components, especially the ICs. Before inserting the ICs into their sockets, check the board for dry joints and solder bridges.

To prevent damage to your system, *always* power off your PAL-1 before installing or removing the PAL-1 Digital Complex Sound Generator board. Ensure that the pins are correctly aligned when inserting the board into the PAL-1 motherboard or when directly connecting it to the PAL-1 expansion port using a 40-pin IDC cable.

Audio output

The PAL-1 Digital Complex Sound Generator incorporates an LM386 audio power amplifier to allow an 8Ω speaker to be directly connected. To connect a speaker or external amplifier use a cable with a 3.5mm mono audio jack. Sound will be heard on one channel only when stereo speakers or a stereo amplifier are used.

Audio volume can be controlled by adjusting the potentiometer (RV1). Refer to *Attenuation* on page 12 for details on setting audio volumes of individual channels. **Note that on power-up the SN76489 registers will contain random values, meaning that random noise may be heard until the attenuation registers are updated.**

I/O address configuration

The SN76489 DCSG has an 8-bit write only data bus, which is memory mapped on the PAL-1 system. The address at which it is mapped can be configured using jumper JP1, as shown in Table 1.1. Note that these addresses are *not* mirorred at $FEEE or $FEEF.

The default address (and the address used in all examples in this manual) is $16EE (5870 decimal). In situations where multiple sound generator boards are installed in the same PAL-1 system, each board must have a unique address.

JP1 pins	Address
1-2	$16EE
2-3	$16EF

Table 1.1: I/O address configuration

2. *Using the PAL-1 Digital Complex Sound Generator*

SN76489 internals

The SN76489 contains 8 internal registers which control the volume and frequencies of each of its three tones and the volume, type and shift-rate of the noise generator. Register values are changed by writing to the SN76489 8-bit data bus, which on the PAL-1 is memory mapped at either $16EE or $16EF.

Each tone generator requires 10 bits of information to set the frequency and 4 bits of information to set the attenuation. Since the SN76489 bus is 8 bits wide, frequency updates require double byte transfers, whilst changing attenuator values is achieved by writing single bytes. When writing data to the SN76489 the most significant bit (msb) of the first byte, the 'latch and data' byte, is always 1. For register updates that require a second byte, a 'data' byte, the msb is always 0.

Latch and data bytes

A 'latch and data' byte is identified by its first bit, which is always 1. This is followed by bits indicating the register (r) to be updated, composed of the destination channel (r0 & r1) and the register type (r2). The final bits (d0 thru d3) are the actual data to be written to the relevant register. When writing to the noise source register, which requires only 3 bits of data, the highest bit (d0) is discarded.

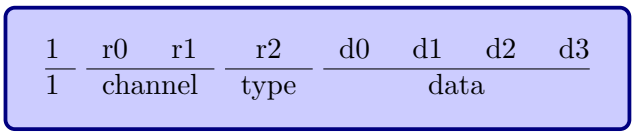


Figure 2.1: Latch and data byte format

Refer to Table 2.1 for details of the channel and type bits (r0-r2).

r0	r1	channel	r2	type
0	0	tone 1	0	tone / noise
0	1	tone 2	1	attenuation
1	0	tone 3		
1	1	noise		

Table 2.1: Latch and data byte register (r) bits

Data bytes

The first bit of a 'data' byte is 0. Since frequency data consists of 10 bits, with the first 4 bits specified in the 'latch and data' byte, only 6 bits of frequency information are required in a 'data' byte (as shown in Figure 2.2).

Data is written to the register most recently latched using a 'latch and data' byte. This means that the lower 6 bits of a frequency can be updated without re-specifying the relevant tone register, for example for frequency sweeps.

Note that data is not ignored when the most recent register latched was something other than a tone frequency register. Rather, the number of bits required for the relevant type will be written to the latched register. For example, audio volume can be faded in or out by consecutive data writes to an attenuation register.

0	x	d0	d1	d2	d3	d4	d5
0	x	data					

Figure 2.2: Data byte format

Calculating frequency values

The SN76489 relies on an external clock signal to generate sounds, which On the PAL-1 Digital Complex Sound Generator board is 2Mhz. The external clock is divided by 16 to arrive at an internal clock of 125Khz.

Each tone and the noise generator has a 10 bit counter and an output bit. When not 0, each counter is decremented at the rate of the internal clock. When a counter reaches 0, the output bit for the relevant tone or the noise generator is flipped from 0 to 1 (or vice versa) and the counter is reset to the value currently in corresponding register. The result is a square wave, with a wavelength of double the register value. Its frequency can be calculated using the following formula

$$f = \frac{N}{32n}$$

where N = reference clock in Hz (i.e. 2Mhz)

n = 10-bit binary counter value

For example, to set tone generator 1 to a frequency of 466.164Hz ($A\sharp_4$), first find I:

$$I = \frac{N}{32f} = \frac{2,000,000}{32 * 466.164} = 134.073$$

Since I must be an integer value, it is rounded down to 134. Next, I must be converted to a 10-bit binary frequency value f as shown in Figure 2.3[1]:

f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
0	0	0	1	0	0	0	1	1	0

Figure 2.3: 446Hz 10 bit frequency value

The 10 bits of frequency data are written to the SN76489 split across 2 bytes. The first byte includes register details (bits r0-r2) and 4 bits of data (f6-f9). The second byte contains the remaining 6 bits of data (f0-f5).

Therefore to set tone generator 2 at 446Hz, the value of the first byte to be written is 166, followed by a second byte with a value of 4 (see Figures 2.4 and 2.5).

Finally, for sound to be heard the attenuation weight of the tone needs to be set. This requires writing a single byte containing the tone register address and attenuation weight. For example, setting the attenuation of tone 2 to 10dB requires a 'latch and data' byte value of 181 (see figure 2.6).

When using BASIC, these values can be written to the SN76489 using `POKE` statements:

```
100 N=5870:REM      SN76489 BASE ADDRESS
110 POKE N,166:REM TONE 2 FREQ REG AND FIRST 4 BITS OF FREQ DATA
120 POKE N,4:REM    REMAINING 6 BITS OF FREQ DATA
130 POKE N,181:REM TONE 2 ATTENUATION REG AND ATTENUATION WEIGHT
```

[1]The example calculations shown here follow the bit hierarchy as used in the original SN76489 documentation as published by Texas Instruments in which the most and least significant bits are transposed.

1	r0	r1	r2	f6	f7	f8	f9
1	0	1	0	0	1	1	0

Figure 2.4: Latch and data byte to set tone 2 at 446Hz

0	x	f0	f1	f2	f3	f4	f5
0	0	0	0	0	1	0	0

Figure 2.5: Data byte with lower bits of 466Hz frequency

1	r0	r1	r2	a0	a1	a2	a3
1	0	1	1	0	1	0	1

Figure 2.6: Latch and data byte setting tone 2 attenuation at 10dB)

Limitations

It takes the SN76489 approximately 32 clock cycles to read data from the bus. In the current implementation the SN76489 READY signal has been left unconnected, which means that it is not possible to use this signal for synchronisation purposes. For reliable operation it is recommened to leave a gap of at least 16μs between writes. This should not be an issue when using high level languages (such as BASIC). However, when using assembly language, care must be taken when writing data to the DCSG to ensure no data is lost.

3. *Sample Code*

Programming the PAL-1 Digital Complex Sound Generator's SN76489 simply requires the ability to write to the mapped I/O address. Many programming languages provide statements to achieve this, including the `POKE` statement in BASIC and `C!` in Forth.

Note that all examples included here use the default I/O address (ie. \$16EE, 5870_{10}).

Further examples may be found on Github at `https://github.com/dimitrit/pal1dcsg`.

BASIC

The following 'tone test' example was taken from the 'Add programmable Sound Effects to Your Computer' article in the July 1982 issue of BYTE magazine[1]. Note that since the PAL-1 uses memory mapped i/o, all `OUT` statements have been replaced with `POKE` statements.

```
100 REM    ****   SN76489A TONE TEST   ****
110 N=5870:REM SOUND GENERATOR BASE ADDRESS
120 POKE N,159:POKE N,191:POKE N,223:POKE N,255:REM CLEAR OUTPUTS
125 REM TONE OUTPUT IS CLOCK/32*I -- ENTER I TO TEST TONE OUTPUT
130 PRINT "DIVIDER␣VALUE":INPUT I
150 B=INT(I/16)
160 A=I-B*16+128
200 POKE N,A:POKE N,B:REM SET 2 BYTE TONE VALUE ON 76489A
220 POKE N,144:REM TURN ON TONE 1 OUTPUT
230 FOR X=0 TO 1000:NEXT X:REM  DELAY
240 GOTO 120
```

Listing 3.1: Tone test

[1] Ciarcia, 60-72.

6502 assembly language

As it is not possible to check whether the SN76489 has finished reading its input buffer, care must be taken to not overflow this buffer when using assembly language. In the example below the PAL-1 RIOT timer is used to wait a number of milliseconds between writes. If the RIOT timer is not available, an appropriate delay must be ensured by other means, for example using `NOP` statements in a loop.

```
;               **** GREENWICH TIME SIGNAL ****
;
; Plays the BBC radio hourly Greenwich time signal pips.
;
; Note that in addition to the PAL-1 Digital Complex Sound Generator card,
; this example requires the RIOT expansion module.
;

sndio           = $16ee                 ; SN76489 i/o address
clk1kt          = $1707                 ; RIOT timer @ 1024T increments

                ; Set attenuation
setattn         .macro c, w             ; c=channel (0-3), w=weight (0-15)
                lda #((((\c<<1)+9)<<4)+\w
                jsr sendbyte
                .endm

                ; Set frequency
setfreq         .macro c, f             ; c=channel (0-3), f=freq (0-1023)
                lda #((((\c<<1)+9)<<4)+(\f&15)
                jsr sendbyte
                lda #(\f>>4)
                jsr sendbyte
                .endm

                ; Delay
delayms         .macro n                ; n=delay in ms
                lda >(\n)
                sta duration
                lda <(\n)
                jsr delay
                .endm

                * = $60

duration        .byte   0               ; delay duration

                * = $200

                .for c:=0, c<4, c+=1    ; silence tones 1-3 and noise
                setattn c, 15
                .next
```

```
                setfreq 0, 63           ; set tone 1 to 1KHz

                ldx #5                  ; play 5 pips
pip             setattn 0, 5            ; set tone 1 attenuation to 10dB
                delayms #100            ; play pip for 100ms
                setattn 0, 15           ; silence tone 1
                delayms #900            ; wait 900ms
                dex
                bne pip                 ; done playing pips?

                setattn 0, 5            ; play final pip for 500ms
                delayms #500            ; wait 500ms
                setattn 0, 15           ; silence tone 1

                brk                     ; and done!

sendbyte        sta sndio               ; (2T + 6T) + 3T send byte to SN76489
                rts                     ; 6T (17us in total incl. lda/jsr)

delay           sta clk1kt              ; wait A milliseconds
delayloop       bit clk1kt
                bpl delayloop           ; RIOT timer not done
delaynext       dec duration            ; ms wait remaining?
                bmi senddone            ; no, all done
                lda #255                ; yes, wait another 255ms
                bne delay               ; and loop again
senddone        rts

                .end
```

Listing 3.2: Greenwich time signal pips

A. *SN76489 Data Formats*

When sending data to the SN76489 the most significant bit (msb) of the 'latch and data' byte (typically the first or only byte written to the SN76489 DCSG) is always 1. Where additional data is required (typically the second byte written), the msb of the 'data' byte is always 0.

The register (r) bits in the 'latch and data' byte determine both the channel being targetted and the type of data being sent (see Table A.1). Once a target has been latched, any subsequent 'data' bytes will affect that target.

r0	r1	channel	r2	type
0	0	tone 1	0	tone / noise
0	1	tone 2	1	attenuation
1	0	tone 3		
1	1	noise		

Table A.1: Latch and data byte register (r) bits

Frequency

Frequency updates require 2 bytes, the first indicating the channel and top 4 bits of frequency data with the remaining frequency data in the second byte. See page 2 for details on calculating frequency data.

1	r0	r1	r2	d6	d7	d8	d9
1	channel		0	data			

Figure A.1: First byte of frequency update

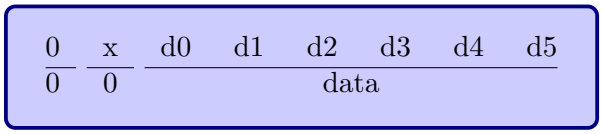


Figure A.2: Second byte of frequency update

Attenuation

Attenuator updates require a single byte, indicating both the channel and attenuation weight (see Table A.2 for weight values).

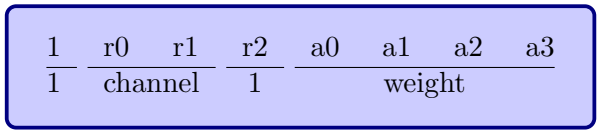


Figure A.3: Attenuator update

a0	a1	a2	a3	weight (dB)
0	0	0	0	0 dB
0	0	0	1	2 dB
0	0	1	0	4 dB
0	1	0	0	8 dB
1	0	0	0	16 dB
1	1	1	1	off

Table A.2: Attenuation weight bits

On power-up the SN76489 registers will contain random values, meaning that random noise may be heard before any attenuation registers are written to.

The following BASIC statements may be used to silence all channels:

```
100 N=5870:REM      SN76489 I/O ADDRESS
110 POKE N,159:REM Tone 1 off
120 POKE N,191:REM Tone 2 off
130 POKE N,223:REM Tone 3 off
140 POKE N,255:REM Noise off
```

Listing A.1: Clear SN76489 attenuation registers

Alternatively, to silence all channels using the KIM monitor:

Press keys	*See printed*
RUB OUT	`KIM`
	`xxxx xx`
1 6 E E	`16EE`
SPACE	`16EE 00`
9 F .	`9F.`
	`16EF xx`
1 6 E E	`16EE`
SPACE	`16EE 00`
B F .	`BF.`
	`16EF xx`
1 6 E E	`16EE`
SPACE	`16EE 00`
D F .	`DF.`
	`16EF xx`
1 6 E E	`16EE`
SPACE	`16EE 00`
F F .	`FF.`
	`16EF xx`

Noise source

Noise source updates require a single byte indicating the noise channel, noise type, and shift rate (see Table A.3 for details on shift rates).

1	r0	r1	r2	x	nt[1]	nf0	nf1
1	1	1	0	0	0 *or* 1	shift rate	

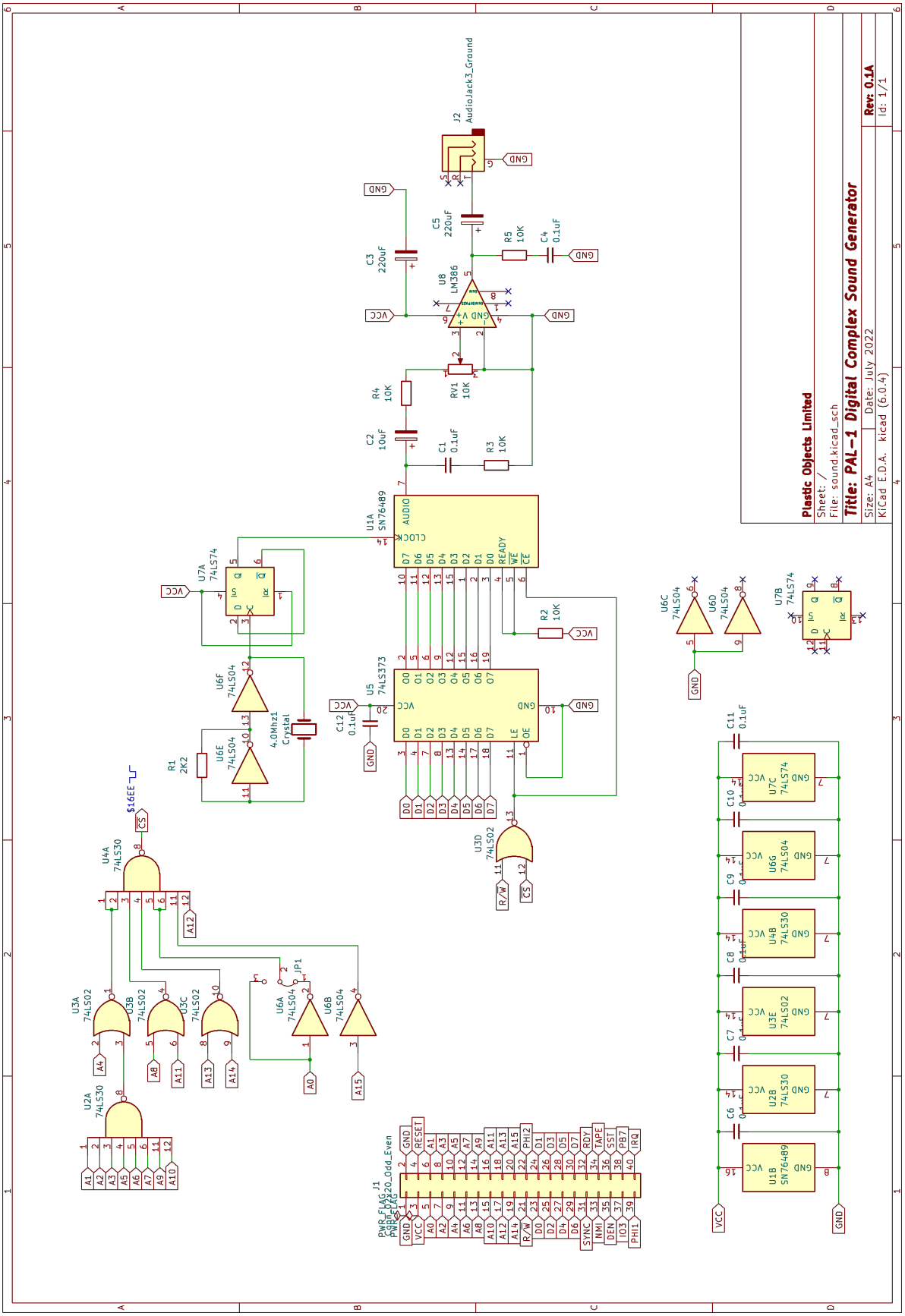
[1] Noise type: 0 = periodic noise, 1 = white noise

Figure A.4: Noise update

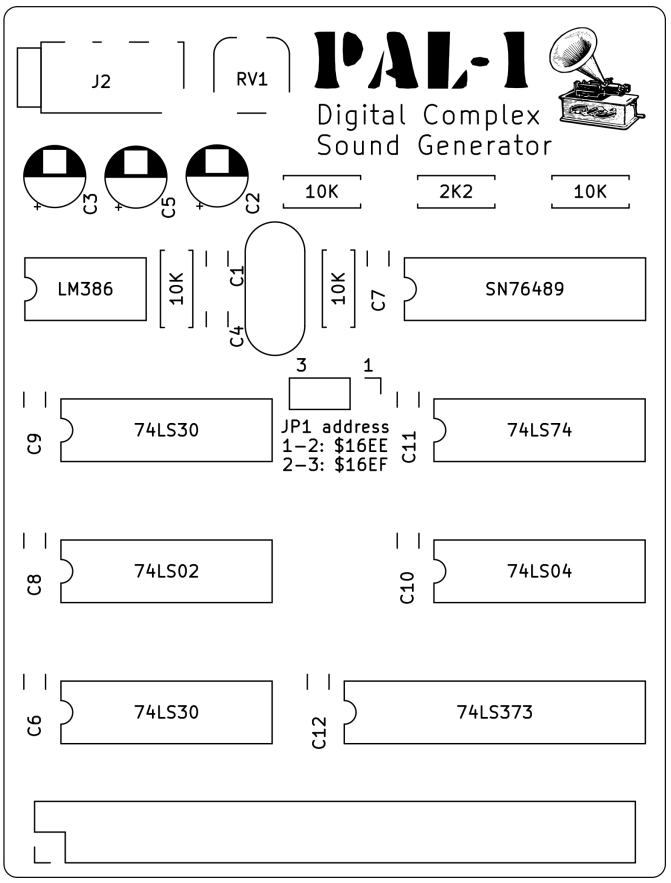
nf0	nf1	shift rate
0	0	n/512
0	1	n/1024
1	0	n/2048
1	1	tone generator 3 output

Table A.3: Shift rates

B. *Schematic Diagram*


Plastic Objects Limited
Sheet: /
File: sound.kicad_sch
Title: PAL-1 Digital Complex Sound Generator
Size: A4
Date: July 2022
KiCad E.D.A. kicad (6.0.4)
Rev: 0.1A
Id: 1/1
J2 AudioJack3_Ground
U8 LM386
U1A SN76489
U5 74LS373
U7A 74LS74
U7B 74LS74
U7C 74LS74
U6E 74LS04
U6F 74LS04
U6C 74LS04
U6D 74LS04
U6G 74LS04
U6A 74LS04
U6B 74LS04
U4A 74LS30
U4B 74LS30
U2A 74LS30
U2B 74LS30
U3A 74LS02
U3B 74LS02
U3C 74LS02
U3D 74LS02
U3E 74LS02
U1B SN76489
4.0Mhz1 Crystal
$16EE
J1 Conn_02x20_Odd_Even
PWR_FLAG
PWR_FLAG

C. *Board layout*



D. *Bill of Materials*

Part	Qty	Value	Description
C1,C4,C6-C12	9	0.1uF	Unpolarized capacitor
C2	1	10uF	Polarized capacitor
C3, C5	2	220uF	Polarized capacitor
J1	1		Connector, double row, 02x20
J2	1		Audio Jack, 3.5mm
JP1	1		Jumper, 3-pole, pins 1+2 closed/bridged
R1	1	2K2	Resistor
R2 - R5	4	10K	Resistor
RV1	1	10K	Potentiometer
U1	1	SN76489	Digital Complex Sound Generator
U2, U4	2	74LS30	8-input NAND
U3	1	74LS02	Quad 2-input NOR gate
U5	1	74LS373	8-bit Latch, 3-state outputs
U6	1	74LS04	Hex Inverter
U7	1	74LS74	Dual D Flip-flop, Set & Reset
U8	1	LM386	Low Voltage Audio Power Amplifier
Y1	1	4.0Mhz	Two pin crystal
PCB	1		PAL-1 Digital Complex Sound Generator PCB

Table D.1: PAL-1 Digital Complex Sound Generator V1.0A components

References

Ciarcia, Steve, 'Add Programmable Sound Effects to Your Computer', *Byte*, 7:7 (July 1982), 60–72.

Ganning, Liu, 'PAL-1 Microcomputer User Manual' (November 2020), ⟨`http://pal.aibs.ws/assets/PAL_en.pdf`⟩ accessed 2022-07-21.

TEXAS INSTRUMENTS Semiconductor Group, The Engineering Staff of, 'SN76489AN' (TEXAS INSTRUMENTS, N.D.).