BuzzKill
Sound Effects Board

User Guide

version 1.0

Last updated February 21, 2025

Architecture

The major functional components of BuzzKill are the oscillators, the envelope generators, the patch module, and the speech generator.

There are eight independent oscillators, designated as OSC0-OSC7. They are subdivided into two groups of four – the modulation oscillators, and the voice oscillators.

The first four oscillators, OSC0-OSC3, are modulation oscillators. The do not produce any sound, but can be used to modify or “modulate” the operation of the voice oscillators. They are designated as MOSC0-MOSC3, with MOSC0 referring to the same oscillator as OSC0, MOSC1 the same as OSC1, etc. The designation OSC is used when referring to the oscillator generically, while MOSC is used when referencing its specific modulation functionality.

The next four oscillators, OSC4-OSC7, are voice oscillators. Their output is directly audible, and each is connected to an envelope generator to control its amplitude. They are designated as VOSC0-VOSC3, with VOSC0 referring to the same oscillator as OSC4, VOSC1 the same as OSC5, etc. The designation OSC is used when referring to the oscillator generically, while VOSC is used when referencing its specific voice functionality.

Each oscillator will continuously produce an 8-bit output value which can be interpreted either as an unsigned number from 0 to 255 or a signed number from -128 to 127 depending on the context. The exact values produced will depend on the parameter settings, such as waveform and frequency, for that specific oscillator.

The four envelope generators are designated as ENV0-ENV3. ENV0 is connected to VOSC0, ENV1 to VOSC1, etc. When activated, each envelope generator will control the amplitude of its associated oscillator’s output, thus adding a dynamic volume contour to the resulting sound. Since the modulation oscillators (MOSC0-MOSC3) do not produce audible output, they do not need envelope generators.

The patch module allows the operation of any of the voice oscillators to be modified by any of the modulation oscillators. The modification can directly affect the output of a voice oscillator, or can alter parameters of a voice oscillator, indirectly affecting its output. Up to five patches can be in use at any time.

The speech module allows for the output of simple human-like sounds for a crude imitation of human speech. The pitch, speed, and several other parameters can be adjusted to produce a variety of different voices or vocal-like sound effects.

Control Interface

BuzzKill can be configured to utilize either an SPI interface or an I^{2}C interface. These will require slightly different hardware connections, but at the software level they are treated identically. Either interface will convert input signals into a series of bytes to be processed as commands and data.

If the I^{2}C interface is utilized, the default I^{2}C address will be 10 (0x0a). A different address can be set at any time and will be retained until overwritten by another address change. If the current address conflicts with another device on the same I^{2}C network, it may be necessary to first disconnect the conflicting device, update the BuzzKill address, and then re-connect the other device.

Command Protocol

All commands are sent as a discrete sequence of bytes. The first byte is a control byte, which may be followed by a specified number of data bytes. The command sequence is not complete until the total expected number of bytes has been received. Control bytes are identified solely by their position (i.e. the first byte of a new sequence) rather than by a particular bit value or pattern. For this reason it is important to always completely send the total number of expected data bytes, before sending another control byte.

The most common command sequence is to set one or more registers to specified values. In this case the control byte consists of the starting register number in bits 2-7 (equivalent to multiplying the register number by four) with bits 0-1 containing a length designator. Lengths of 1-3 bytes can be accommodated in this way; for greater lengths bits 0-2 should be set to zeroes and a separate byte should follow which designates the length (i.e. the number of data bytes to follow).

Examples command sequences for setting registers:

```
0x15 0x8f                        Set register 5 to value 0x8f (0x15=5*4+1)
0x16 0x81 0x82                   Set register 5 to 0x81 and register 6 to 0x82
0x80 0x04 0x12 0x15 0x76 0xae    Set registers 32-35 to values 0x12, 0x15, 0x76, 0xae
```

A zero length designation (00 in control bits 0-1 followed by a separate 0 byte) has a special meaning. It will reset all registers from the specified register forward. For example:

```
0x00 0x00                        Reset all registers to default values
0x80 0x00                        Reset registers 32-59 to default values
```

Some commands either do not need any following data bytes, or always need a fixed number of following data bytes. For these there is no need to specify any length information; the length is already determined by the specific command. For example the command byte 0xfb (Enter Sleep Mode) is self-contained and needs no following bytes, whereas the command byte 0xf9 (Set Custom Waveform) always expects exactly 256 bytes to follow.

See the following chart for detailed command information.

Command Summary

Control Value	*Description*	*Format Details*
0 (0x00) – 239 (0xef)	Set register value(s)	Bits 0-1: Data length (number of data bytes) Bits 2-7: Starting register number For lengths>3, set low bits to “00” and send a separate additional byte containing length Follow by specified number of data bytes If final length=0, reset from starting register forward
240 (0xf0) – 243 (0xf3)	Load speech buffer	Bits 0-1: Data length (number of data bytes) For lengths>3, set low bits to “00” and send a separate additional byte containing length Follow by specified number of data bytes If final length=0, clear speech buffer
244 (0xf4)	Set speech speed	Preset data length of 1: Follow by 1 byte containing desired value
245 (0xf5)	Set speech parameters (see Speech section)	Preset data length of 8: Follow by 8 bytes containing desired values
246 (0xf6)	Stop speaking	Single-byte command: No following data bytes
247 (0xf7)	Start speaking	Single-byte command: No following data bytes
248 (0xf8)	Reset Oscillator(s)	Preset data length of 1: Follow by 1 byte (bit field indicating which oscillators to reset)
249 (0xf9)	Set custom waveform values	Preset data length of 256: Follow by 256 bytes containing desired values
250 (0xfa)	Set new I^{2}C address	Preset data length of 3: Follow by NEWADDR, Follow by NEWADDR XOR 0b01010101, Follow by NEWADDR XOR 0b10101010
251 (0xfb)	Enter sleep mode	Single-byte command: No following data bytes
252 (0xfc) – 254 (0xfe)	Reserved	
255 (0xff)	No operation (ignored)	Single-byte command: No following data bytes

Control Registers

Core chip functions are controlled by sixty 8-bit registers. The values contained in these registers determine the frequencies and waveforms of each oscillator, as well as volume levels, patches, and other special features. Modifying the values in various registers is the primary method of controlling various aspects of sound output.

Registers are numbered from 0-59. They are referred to strictly by their number when issuing commands that change their values. For example to change the waveform of OSC0 you would issue a command targeting register number 3 (see Command Protocol above and Register Summary below).

Each register is assigned a specific function, or a set of related functions. For example, setting the master volume level. If you wish to change the master volume level, you do so by writing a new value to the appropriate register number (48).

See the following chart for full details on each register and what it controls. For ease of reference each function is marked by a three-letter tag which will be defined in the appropriate following documentation section.

Many registers may appear to be “duplicates” in the chart because they use the same function tags. However, each register is serving that function for a different oscillator, envelope, etc. For example, each of the eight oscillators has a WAV setting, so there will be a total of eight registers containing the tag “WAV”. The grouping information in the far-right column should make clear which register belongs to which device.

Some functions need more than 8 bits and so are spread across registers. FRQ (Frequency) for example needs 16 bits total and thus uses two registers, one for the lower 8 bits and one for the upper 8 bits.

Other functions need fewer than 8 bits and therefore can share a register with other functions. In these cases you can use the tags to determine which bits in the register byte are dedicated to that function. Some familiarity with bitwise operations will be helpful here.

Note that there is no way to alter only certain bits within a register. The entire 8-bit value must be written at once. If you need to modify certain bits and leave others unchanged, you will need to include the proper values for the unchanged bits in your new data byte.

Register Example

If you wish to set the frequency of VOSC0 to a value of 0x2548, the correct command would be:

```
0x42 0x48 0x25
```

Please review the Command Protocol section above if that doesn’t make sense.

Register Summary

	Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*	*Default*	*Group*
0	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC0 (MOSC0)
1	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
2	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
3	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
4	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC1 (MOSC1)
5	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
6	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
7	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
8	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC2 (MOSC2)
9	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
10	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
11	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
12	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC3 (MOSC3)
13	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
14	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
15	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
16	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC4 (VOSC0)
17	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
18	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
19	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
20	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC5 (VOSC1)
21	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
22	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
23	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
24	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC6 (VOSC2)
25	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
26	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
27	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	
28	FRQ_7	FRQ_6	FRQ_5	FRQ_4	FRQ_3	FRQ_2	FRQ_1	FRQ_0	0x00	OSC7 (VOSC3)
29	FRQ_{15}	FRQ_{14}	FRQ_{13}	FRQ_{12}	FRQ_{11}	FRQ_{10}	FRQ_9	FRQ_8	0x00	
30	MID_7	MID_6	MID_5	MID_4	MID_3	MID_2	MID_1	MID_0	0x80	
31	WAV_2	WAV_1	WAV_0	REV	INV	STP_2	STP_1	STP_0	0x00	

Register Summary (continued)

	Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*	*Default*	*Group*
32	CRV_1	CRV_0	RLR_1	RLR_0	DCR_1	DCR_0	ATR_1	ATR_0	0x00	ENV0
33	DEC_3	DEC_2	DEC_1	DEC_0	ATT_3	ATT_2	ATT_1	ATT_0	0x00	
34	GAT	SUS_6	SUS_5	SUS_4	SUS_3	SUS_2	SUS_1	SUS_0	0x7f	
35	MIX_3	MIX_2	MIX_1	MIX_0	REL_3	REL_2	REL_1	REL_0	0xf0	
36	CRV_1	CRV_0	RLR_1	RLR_0	DCR_1	DCR_0	ATR_1	ATR_0	0x00	ENV1
37	DEC_3	DEC_2	DEC_1	DEC_0	ATT_3	ATT_2	ATT_1	ATT_0	0x00	
38	GAT	SUS_6	SUS_5	SUS_4	SUS_3	SUS_2	SUS_1	SUS_0	0x7f	
39	MIX_3	MIX_2	MIX_1	MIX_0	REL_3	REL_2	REL_1	REL_0	0xf0	
40	CRV_1	CRV_0	RLR_1	RLR_0	DCR_1	DCR_0	ATR_1	ATR_0	0x00	ENV2
41	DEC_3	DEC_2	DEC_1	DEC_0	ATT_3	ATT_2	ATT_1	ATT_0	0x00	
42	GAT	SUS_6	SUS_5	SUS_4	SUS_3	SUS_2	SUS_1	SUS_0	0x7f	
43	MIX_3	MIX_2	MIX_1	MIX_0	REL_3	REL_2	REL_1	REL_0	0xf0	
44	CRV_1	CRV_0	RLR_1	RLR_0	DCR_1	DCR_0	ATR_1	ATR_0	0x00	ENV3
45	DEC_3	DEC_2	DEC_1	DEC_0	ATT_3	ATT_2	ATT_1	ATT_0	0x00	
46	GAT	SUS_6	SUS_5	SUS_4	SUS_3	SUS_2	SUS_1	SUS_0	0x7f	
47	MIX_3	MIX_2	MIX_1	MIX_0	REL_3	REL_2	REL_1	REL_0	0xf0	
48	VOL_3	VOL_2	VOL_1	VOL_0	ENA_3	ENA_2	ENA_1	ENA_0	0xf0	GLOBAL
49	HLT_7	HLT_6	HLT_5	HLT_4	HLT_3	HLT_2	HLT_1	HLT_0	0x00	
50	PDN_1	PDN_0	PSC_1	PSC_0	PTY_3	PTY_2	PTY_1	PTY_0	0x00	PATCH0
51	PPM_7	PPM_6	PPM_5	PPM_4	PPM_3	PPM_2	PPM_1	PPM_0	0x00	
52	PDN_1	PDN_0	PSC_1	PSC_0	PTY_3	PTY_2	PTY_1	PTY_0	0x00	PATCH1
53	PPM_7	PPM_6	PPM_5	PPM_4	PPM_3	PPM_2	PPM_1	PPM_0	0x00	
54	PDN_1	PDN_0	PSC_1	PSC_0	PTY_3	PTY_2	PTY_1	PTY_0	0x00	PATCH2
55	PPM_7	PPM_6	PPM_5	PPM_4	PPM_3	PPM_2	PPM_1	PPM_0	0x00	
56	PDN_1	PDN_0	PSC_1	PSC_0	PTY_3	PTY_2	PTY_1	PTY_0	0x00	PATCH3
57	PPM_7	PPM_6	PPM_5	PPM_4	PPM_3	PPM_2	PPM_1	PPM_0	0x00	
58	PDN_1	PDN_0	PSC_1	PSC_0	PTY_3	PTY_2	PTY_1	PTY_0	0x00	PATCH4
59	PPM_7	PPM_6	PPM_5	PPM_4	PPM_3	PPM_2	PPM_1	PPM_0	0x00	

Waveforms

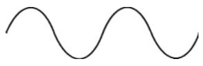
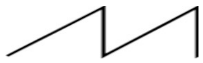
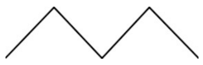
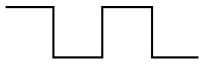
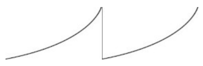
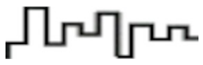
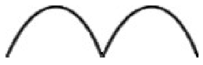
Each oscillator waveform is fully specified by 32 bits spread across 4 registers. The bits are grouped according to the following functions.

Frequency (FRQ)

Each oscillator operates at a frequency defined by the FRQ bits. The complete 16-bit value represents the frequency in steps of 1/16 Hz. Alternatively, you can think of the bits FRQ_4-FRQ_{15} as representing the integer portion of the value, while FRQ_0-FRQ_3 represents the fractional portion. The maximum possible frequency is thus 4095.9375 Hz.

Waveform Shape (WAV)

For each oscillator there are seven pre-defined waveform shapes and one custom-programmable shape. Select the shape by setting the WAV bits in the appropriate control register as follows:

Number	*Bit Pattern*	*Name*	*Shape*	*Description*
0	000	Sine		The most "pure" sound, very smooth and clean with no extra harmonics.
1	001	Ramp/Sawtooth		A "buzzy" sound with many harmonics. Can resemble stringed instruments.
2	010	Triangle		Some harmonics, smoother than a ramp but harsher than a sine. Can resemble a piano or xylophone.
3	011	Pulse		Some weaker harmonics. Can sound "electronic" or like reed instruments.
4	100	Exponential		Similar to a ramp, more useful for modulations than direct audio usage.
5	101	Noise		Useful for drums, gunshots, crowds cheering, and wind sounds.
6	110	Custom		User defined, so can sound like anything.
7	111	Hilltop		Similar to a triangle, more useful for modulations than direct audio usage.

Midpoint (MID)

Each oscillator has a midpoint setting in the corresponding MID bits. The midpoint represents the point in time (on a scale of 0-255) at which half the waveform has been output. The second half of the waveform will then be output in the remaining time, scaled appropriately.

The default setting is 0x80, where the time midpoint corresponds with the waveform midpoint. This is the “normal” setting at which the waveform output will appear in its “regular” form (a regular sine wave, regular triangle, etc.). Reducing the midpoint setting will shift the output so that the first half of the waveform is compressed while the second half is stretched, while increasing the setting will stretch the first half and compress the second half.

Although it works with all waveform shapes, the most common use for the midpoint setting is with pulse waveforms. In this case it controls the point at which the waveform transitions from high to low, making it identical to what other systems might call “pulse width” or “duty cycle”. For example, a midpoint setting of 0x80 produces a square wave, 0x40 is a 25% wave, and 0xc0 is a 75% wave.

Reverse (REV)

Any waveform can be reversed by setting the appropriate register REV bit. A reversed waveform is mirrored horizontally, so for example a ramp waveform will start high and slowly fall, while a triangle waveform will appear unchanged. Reversing a waveform will generally make no audible difference, but can be useful in patches.

Invert (INV)

Any waveform can be inverted by setting the appropriate register INV bit. An inverted waveform is mirrored vertically, so for example both ramp and triangle waveforms will start high and move lower. Inverting a waveform will generally make no audible difference, but can be useful in patches.

Step Size (STP)

Normally waveforms are updated by the smallest possible amount during each output frame, resulting in the smoothest output possible. Setting a step value in the appropriate STP bitfield will reduce the effective resolution of the waveform value, creating a more “jagged” output. The STP value (0-7) represents how many bits to remove from the normally 8-bit output value, so for example at STP=4 the output will be restricted to only 16 values, while at STP=7 the output will be restricted to only 2 values.

Envelopes

Each Voice Oscillator (VOSC) has an associated Envelope Generator which dynamically controls its output amplitude as a function of time. Specifically, the generator is an ADSR envelope generator, short for Attack/Decay/Sustain/Release. Each generator also has an associated Gate to control the ADSR sequence.

Gate (GAT)

The envelope gate controls the start and end of the envelope sequence. Turning the gate on (changing the GAT bit from 0 to 1) designates a new note and triggers a new attack mode. Turning the gate off (changing the GAT bit from 1 to 0) designates the end of a note and triggers a release mode. Using a piano as an example, turning the gate on is equivalent to pressing a key, and turning the gate off is equivalent to releasing the key.

Attack (ATT/ATR)

Once gated on, a note builds up to maximum amplitude over a specified period of time. The higher the attack setting, the longer it takes the note to reach maximum amplitude. This can range from zero to a little over five seconds.

The complete attack time is controlled by a combination of two settings. ATR (Attack Range) determines the general scale of values from a choice of four ranges. ATT then selects the specific value within that range. See the following tables for specifics.

Decay (DEC/DCR)

Once a note reaches maximum amplitude, the Attack phase ends and the Decay phase begins. In this phase, the amplitudes begins to fall toward the Sustain level (see below). The Decay time will determine how long it takes to reach the new level. It uses two settings in the same way as Attack, in this case DCR for general scale and DEC for specific value.

For simplicity, Decay uses the same scale and rates as Attack. Note however that the Decay phase may not last the same amount of time as the Attack phase even if both are set to the same rate. This is because Attack always moves between zero and maximum amplitudes, while Decay moves between maximum and the Sustain level, which may be well above zero.

Sustain (SUS)

Sustain is simply the amplitude level that will be held as long as the gate remains on. It is specified as a proportion in the range 0-127. With a setting of 64 the amplitude will hold at approximately half its maximum level, while a setting of 127 will maintain maximum amplitude until the note is stopped.

Release (REL/RCR)

When a note is “gated off” the final envelope phase begins, as the amplitude then begins to fall away to zero level. How quickly it fades to zero is determined by the Release setting. Like Attack and Decay, it uses a combination of two settings, in this case RLR for general scale and REL for specific value. Like Decay, Release is specified on the same scale although it may actually end somewhat earlier depending on the Sustain level.

Attack/Decay/Release Rates

These tables give the specific time values for each combination of Range Select bits (ATR, DCR, or RLR) and Rate select bits (ATT, DEC, or REL).

The time values are in milliseconds, and represent the total time needed for a “full swing” from zero to maximum amplitude (or vice versa). For Decay/Release, the rate of change will be identical but total time may differ because of reduced amplitude swing.

Range Select 0 (Bits=00)		
Rate	*Bits*	*Time*
0	0000	0.0
1	0001	7.7
2	0010	14.4
3	0011	23.1
4	0100	30.8
5	0101	38.5
6	0110	46.2
7	0111	53.9
8	1000	61.6
9	1001	69.3
10	1010	77.0
11	1011	84.7
12	1100	92.4
13	1101	100.1
14	1110	107.8
15	1111	115.5

Range Select 1 (Bits=01)		
Rate	*Bits*	*Time*
0	0000	123.0
1	0001	143.5
2	0010	164.0
3	0011	184.5
4	0100	205.0
5	0101	225.5
6	0110	246.0
7	0111	266.5
8	1000	287.0
9	1001	307.5
10	1010	328.0
11	1011	348.5
12	1100	369.0
13	1101	389.5
14	1110	410.0
15	1111	430.5

Range Select 2 (Bits=10)		
Rate	*Bits*	*Time*
0	0000	492.8
1	0001	554.4
2	0010	616.0
3	0011	677.6
4	0100	739.2
5	0101	800.8
6	0110	862.4
7	0111	924.0
8	1000	985.6
9	1001	1047.2
10	1010	1108.8
11	1011	1170.4
12	1100	1232.0
13	1101	1293.6
14	1110	1355.2
15	1111	1416.8

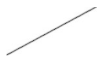
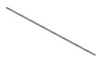
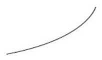
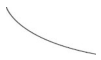
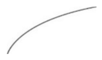
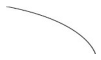
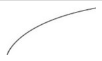
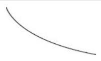
Range Select 3 (Bits=11)		
Rate	*Bits*	*Time*
0	0000	1478.4
1	0001	1724.8
2	0010	1971.2
3	0011	2217.6
4	0100	2464.0
5	0101	2710.4
6	0110	2956.8
7	0111	3203.2
8	1000	3449.6
9	1001	3696.0
10	1010	3942.4
11	1011	4188.8
12	1100	4435.2
13	1101	4681.6
14	1110	4928.0
15	1111	5174.4

Envelope Curve (CRV)

Amplitudes do not have to increase/decrease at a constant rate throughout the entire phase. If they do, they are called linear. However most naturally-occurring sounds follow an exponential curve, starting quickly and slowing as the maximum/minimum level is approached. The opposite can also be useful for sound effects, starting slowly and increasing over time.

The curve of each envelope is controlled by the two corresponding CRV bits, according to the table at right.

Unfortunately there is no standard for naming these curve options, except for linear. The terms “exponential” and “logarithmic” are often used but with no universal agreement as to which term applies to which specific shape.

Bits	*Attack Shape*	*Decay/Release Shape*
00		
01		
10		
11		

Mix Volume (MIX)

The outputs from all enabled envelopes are combined (mixed) to form a single total output. This mixing is done proportionally according to each envelope’s MIX setting. This allows different voices to have different levels of prominence in the final output. Note that a value of zero represent a minimum level of contribution, but not a complete absence of sound (if you don’t want a voice to be heard at all, see Enable in Global Settings). A general rule-of-thumb is for the sum of all MIX settings for active voices should total about 15. So for two active voices, a MIX setting of 7 for both envelopes will result in an equal combination of each waveform. Setting one value to 10 and the other to 5 will result in one voice seeming about twice as “loud” in the final output.

Global Settings

Volume (VOL)

This setting controls the overall volume level for the final speaker output. It is applied after mixing of individual voices is completed according to the MIX settings. The values can range from 0-15, but note that zero does not represent “off” or an absence of sound, but rather the minimum possible level of output.

Enable (ENA)

These four bits control which voice oscillator outputs will appear in the final output mix. Each voice has a corresponding Enable bit – ENA_3 enables VOSC3, ENA_2 enables VOSC2, ENA_1 enables VOSC1, and ENA_0 enables VOSC0. Setting the proper ENA bit allows the oscillator’s envelope output to be audible (at a level determined by its MIX setting), whereas clearing the ENA bit will block the oscillator’s envelope output and render it inaudible. Note that this only affects the audible output of each oscillator; all oscillators will continue to run normally whether audible or not.

Halt (HLT)

Each oscillator (OSC0-OSC7) has a corresponding halt bit (HLT_0-HLT_7). Setting the halt bit will reset the oscillator to the start of its cycle and hold it there as long as the bit remains set. The oscillator value during this period will remain fixed at its initial value according to the selected waveform. Once the halt bit is cleared, the oscillator will resume running and updating normally.

Patches

Patches allow the output of a modulation oscillator to modify the operation of a voice oscillator. Up to five patches may be used, and each patch can be one of fifteen types.

Each patch is specified by two registers. One register specifies the patch type as well as the “source” oscillator (one of the modulation oscillators) and “destination” oscillator (one of the voice oscillators). The other register provides an optional parameter whose meaning will vary according to the patch type.

For each patch, the type is specified by the four PTY bits. The source oscillator is specified by the two PSC bits which will refer to one of the modulation oscillators (MOSC0-MOSC3, or OSC0-OSC3), while the destination oscillator is specified by the two PDN bits which will refer to one of the voice oscillators (VOSC0-VOSC3, or OSC4-OSC7). If a parameter is needed, it is provided by the eight PPM bits.

“Multi” Patches

While most patch types affect the output of only one oscillator (the one specified by the PDN bits), several of the patches will affect the output of three separate oscillators. These are referred to as multi-patches. They work essentially like three separate patches specifying the same source oscillator but three different destination oscillators. However with a multi-patch the three destination oscillators are assumed to be contiguous, starting with the oscillator specified by the PDN bits and continuing automatically with the next two. Thus with PDN=00 the affected oscillators will be VOSC0, VOSC1, and VOSC2. With PDN=01 the affected oscillators will be VOSC1, VOSC2, and VOSC3. PDN settings of 10 or 11 with a multi-patch are undefined and should not be used.

Patch type zero (PTY=0000) means that the patch is inactive. The other patch types are described below.

Patch Type 1 (PTY=0001): Frequency Scale

This patch will scale the frequency of the destination oscillator according to the current value of the source oscillator. The parameter provides the scaling factor and should be in the range 1-255.

Scaling means that the frequency of the destination oscillator will be reduced by a percentage determined by current value of the source oscillator, considered as an unsigned number from 0 to 255. When the source oscillator value is near zero, the destination oscillator frequency will be near its normal level. When the source oscillator value is at a higher value, the destination oscillator frequency will be lowered proportionally according to the scaling factor. The frequency will never be adjusted above its normal set level, only downward, and will never be adjusted below zero.

The larger the scaling factor, the larger the effect on the frequency. With scaling set at maximum (255), a full “swing” of the source oscillator value will result in the frequency varying between approximately its set value and zero. With scaling set at half (128), a full swing of the source oscillator value will result in the frequency varying between approximately its set value and one-half its set value.

The exact formula for the frequency reduction (as a proportion of the set frequency) is:

```
({source oscillator value} x {scaling factor} / 256 + 1) / 256
```

Patch Type 2 (PTY=0010): Frequency Shift

This patch will shift the frequency of the destination oscillator according to the current value of the source oscillator. The parameter provides the scaling factor and should be in the range 1-255.

Shifting means that the frequency of the destination oscillator will be adjusted upward or downward by an amount determined by the current value of the source oscillator, considered as a signed number from -128 to 127. When the source oscillator value is below zero, the frequency will be adjusted downward. When the source oscillator value is above zero, the frequency will be adjusted upward.

The larger the scaling factor, the larger the effect on the frequency. A scaling factor of 255 will produce a swing range of over 2000 Hz both upward and downward. A scaling factor of 128 would produce a swing range of just over 1000 Hz. Note that with large scaling factors it is possible to “wrap” the frequency value around the upper/lower limits, in which case the value will be treated modulus 4096. Thus a value above 4095 will be heard as a very low frequency, while a value below zero will be heard as a very high frequency. This can lead to odd and jarring sounds, which may or may not be what you want, so choose the scaling factor accordingly.

Patch Type 3 (PTY=0011): Midpoint Shift

This patch will shift the midpoint of the destination oscillator according to the current value of the source oscillator. The parameter provides the scaling factor and should be in the range 1-255.

Shifting means that the midpoint of the destination oscillator will be adjusted upward or downward by an amount determined by the current value of the source oscillator, considered as a signed number from -128 to 127. When the source oscillator value is below zero, the midpoint will be adjusted downward. When the source oscillator value is above zero, the midpoint will be adjusted upward.

The larger the scaling factor, the larger the effect on the midpoint. A scaling factor of 255 will produce a swing range of 126 upward and 127 downward. A scaling factor of 128 would produce a swing range of 63 upward and 64 downward. Note that with large scaling factors it is possible to “wrap” the midpoint value beyond the normal 0-255 range, in which case the value will be treated modulus 256.

Patch Type 4 (PTY=0100): Amplitude Scale

This patch will scale the amplitude of the destination oscillator's envelope according to the current value of the source oscillator. The parameter provides the scaling factor and should be in the range 1-255.

Scaling means that the amplitude of the destination oscillator will be reduced by a percentage determined by current value of the source oscillator, considered as an unsigned number from 0 to 255. When the source oscillator value is near zero, the destination oscillator amplitude will also be near zero. When the source oscillator value is at a higher value, the destination oscillator amplitude will be proportionally higher according to the scaling factor. The adjusted amplitude will always be from 0% to 100% of the normal amplitude, so the amplitude will never be adjusted above its normal level.

The larger the scaling factor, the closer the maximum adjusted amplitude will be to the normal amplitude. With scaling set at maximum (255), a full "swing" of the source oscillator value will result in the adjusted amplitude varying between 0%-100% of the normal amplitude. With scaling set at half (128), a full swing of the source oscillator value will result in the adjusted amplitude varying between 0%-50% of the normal amplitude.

The exact formula for the adjusted amplitude is:

```
{source oscillator value} x {scaling factor} x {normal amplitude + 1} / 65536
```

Patch Type 5 (PTY=0101): Amplitude Level

This patch will set the amplitude of the destination oscillator's envelope according to the current value of the source oscillator. The parameter provides a signed scaling factor and should be in the range 1 to 127 or -1 to -127.

The amplitude is not shifted or scaled in any way based on the normal amplitude level. Rather the new amplitude level is set directly based on the source oscillator value, ignoring and overwriting the normal amplitude level of the destination oscillator envelope.

The new amplitude is determined by the current value of the source oscillator, considered as an unsigned number from 0 to 255. The magnitude of the scaling factor determines the range of the amplitude, while the sign of the scaling factor determines if the range is expressed upward (from the minimum) or downward (from the maximum).

The larger the magnitude of the scaling factor, the more the amplitude level will vary as the source oscillator moves through its complete range. With scaling set at ±127, a full "swing" of the source oscillator value will result in the destination amplitude also moving through its maximum possible range. With scaling set at ±64, a full swing of the source oscillator value will result in the amplitude moving through half its possible range.

When the source oscillator value is near zero, the destination oscillator amplitude will be near zero if the scaling factor is positive, and near maximum if the scaling factor is negative. When the source oscillator is at a higher value, the destination amplitude will be proportionally higher if the scaling factor is positive, and lower if the scaling factor is negative.

Patch Type 6 (PTY=0110): Envelope Gate

This patch will set the gate of the destination oscillator envelope either on or off based on the current value of the source oscillator. The parameter provides the threshold and should be in the range 1-255.

The current value of the source oscillator is compared against the threshold, considering both as unsigned numbers from 0-255. If the oscillator value is lower than the threshold, the gate is turned off (GAT bit is set to 0). Otherwise the gate is turned on (GAT bit set to 1).

Patch Type 7 (PTY=0111): Hard Sync

This patch will reset the destination oscillator each time the source oscillator begins a new cycle. This patch does not utilize a parameter.

This patch will force the destination oscillator into a fixed cycle relationship with the source oscillator, even if the two are operating at different frequencies. In synthesizer terminology this is often referred to as “hard” sync.

Patch Type 8 (PTY=1000): Soft Sync

This patch will reverse, invert, or reset the destination oscillator based on the current value of the source oscillator, considered it as a signed number in the range -128 to 127. The parameter is considered as a bit field and provides flags for selecting the desired action(s).

This patch can implement a form of “soft” sync. Although the term is not well-defined, it generally refers to any method of synchronizing one oscillator to another that does not involve aligning the cycle starting points.

Parameter bit 4 (PPM_4) selects the Reverse action. If this bit is set to 1, the destination oscillator’s waveform will be reversed (mirrored left-to-right) whenever the source oscillator’s value is positive.

Parameter bit 3 (PPM_3) selects the Invert action. If this bit is set to 1, the destination oscillator’s waveform will be inverted (mirrored top-to-bottom) whenever the source oscillator’s value is positive.

Parameter bit 0 (PPM_0) selects the Reset action. If this bit is set to 1, the destination oscillator will be reset whenever the source oscillator’s value crosses from negative to positive. Note that although this action is similar to hard sync, it is distinct because the reset need not occur at the beginning of the source oscillator’s cycle and may even occur multiple times within one cycle.

Patch Type 9 (PTY=1001): Ring Modulation

This patch replaces the normal value of the destination oscillator with the product of that value and the value of the source oscillator, considering both values as signed numbers in the range -128 to 127. This patch does not utilize a parameter.

Patch Type 10 (PTY=1010): Amplitude Scale Multi

This patch operates in the same way as Patch Type 4, except as a multi-patch. See previous description of “multi” patches and Patch Type 4 for details.

Patch Type 11 (PTY=1011): Amplitude Level Multi

This patch operates in the same way as Patch Type 5, except as a multi-patch. See previous description of “multi” patches and Patch Type 5 for details.

Patch Type 12 (PTY=1100): Envelope Gate Multi

This patch operates in the same way as Patch Type 6, except as a multi-patch. See previous description of “multi” patches and Patch Type 6 for details.

Patch Type 13 (PTY=1101): Hard Sync Multi

This patch operates in the same way as Patch Type 7, except as a multi-patch. See previous description of “multi” patches and Patch Type 7 for details.

Patch Type 14 (PTY=1110): Soft Sync Multi

This patch operates in the same way as Patch Type 8, except as a multi-patch. See previous description of “multi” patches and Patch Type 8 for details.

Patch Type 15 (PTY=1111): Output Pin

This patch will set the POUT output pin either high or low based on the current value of the source oscillator. The parameter provides the threshold and should be in the range 1-255.

The current value of the source oscillator is compared against the threshold, considering both as unsigned numbers from 0-255. If the oscillator value is lower than the threshold, POUT is set low. Otherwise POUT is set high.

Note that once this patch has been applied, removing it will leave POUT in whatever state it was last set.

Speech Mode

In speech mode, the frequencies and amplitudes of the four voice oscillators are automatically controlled to produce speech-like sounds. The envelope settings (except MIX) are ignored and the amplitude for each oscillator is instead directly set by the speech module, but patches can still alter the final amplitude. Other oscillator registers function normally, but the FRQ registers will be continually overwritten by the speech module.

The adjustments are based on a list of phoneme tokens which have been entered in advance. The list can be as long as 255 tokens, and speech mode automatically terminates when the end of the list is reached. Each token represents a specific speech sound, which can be combined to form words. The particular sounds available are largely based on English, but can also be utilized to some extent for other languages.

To create a phrase for speaking, a data string should be created consisting of the numeric values of each token in the phrase, one byte per token. Each token has a two-letter "tag" for ease of reference, but the numeric value is what is actually used in the data string. See the following chart for the complete list of tokens and their sound values.

Determining the correct phonemes can usually be done by carefully sounding out the desired word and finding the matching sounds from the examples in the chart. Alternatively, the IPA (International Phonetic Alphabet) symbols are also included and can be matched to the pronunciation guides available in most dictionaries.

Although most tokens have a single corresponding IPA symbol, several have a double symbol. These represent "diphthongs," which are two sounds commonly combined into a single sequence. If a diphthong is available for a pair of sounds, it should be preferred over using two separate tokens. For example a common symbol pair is "ʊə". Each component is an available token (token UH=ʊ, token AX=ə), but preference should be given to the token UR which represents both symbols together.

The available phonemes are meant to represent all the sounds present in general English rather than any particular dialect. Most varieties of English will not contain all these phonemes, and some of the indicated sounds may seem confusing if they are not part of your dialect or accent. For example many speakers do not distinguish the word "whale" from the word "wail," so the inclusion of both a W* token and a WH token may seem strange. Many Americans do not distinguish the first vowel sound in "father" from the first vowel sound in "bother" so the AU token may seem redundant. Fewer still would distinguish H* from X*, but some do so both phonemes are available.

There are also a few tokens that do not represent independent sounds but rather "weaker" versions of another phoneme. These can be useful at the end of words, where speakers often fail to fully enunciate a final consonant. They can also be useful for glides, where a vowel sound moves toward a final W or Y sound, implied but not fully articulated.

Speech Phoneme Tokens

Dec	*Hex*	*Tag*	*IPA*	*Examples / Notes*
0	00	OW	oʊ	low, boat, nose
1	01	AW	aʊ	how, now, shout
2	02	EY	eɪ	made, pay, weigh
3	03	AI	eə	chair, there, wear
4	04	AY	aɪ	high, sky, pie
5	05	EA	ɪə	ear, here, beer
6	06	OY	ɔɪ	boy, join, coil
7	07	UR	ʊə	cure, tour, fury
8	08	AE	æ	cat, plaid, laugh
9	09	AA	ɑ	father, pot, balm
10	0a	AU	ɒ	bother, sock, honest
11	0b	EH	ɛ	bread, said, many
12	0c	IY	i	bee, meat, ski
13	0d	AO	ɔ	ford, abort, caught
14	0e	ER	ɝ	bird, term, pearl
15	0f	AH	ʌ	bug, monkey, double
16	10	UW	u	who, loot, blue
17	11	UH	ʊ	look, wolf, bush
18	12	IH	ɪ	pin, gym, busy
19	13	AX	ə	gallon, pencil, comma
20	14	S*	s	sit, less, circle
21	15	SH	ʃ	fish, ocean, machine
22	16	F*	f	fat, phone, enough
23	17	V*	v	seven, vine, of
24	18	Z*	z	zoo, buzz, his
25	19	ZH	ʒ	pleasure, division, azure
26	1a	TH	θ	thin, thank, ether
27	1b	DH	ð	then, leather, either
28	1c	M*	m	man, summer, comb
29	1d	N*	n	net, funny, know
30	1e	NG	ŋ	sing, pink, tongue
31	1f	H*	h	head, hind, hay
32	20	X*	ħ	ahead, behind, Hanukkah
33	21	R*	ɹ	red, carrot, wrench
34	22	RX	ɹ̣	nurse, dear, break
35	23	L*	l	live, release, allow
36	24	LX	ɫ	ball, help, able
37	25	W*	w	away, wit, sway
38	26	WH	ʍ	whale, white, whack
39	27	Y*	j	you, yellow, onion
40	28	WX		WX is a weaker version of W
41	29	YX		YX is a weaker version of Y
42	2a	KX		KX is a weaker version of K
43	2b	GX		GX is a weaker version of G
44	2c	T*	t	talk, matter, ripped
45	2d	D*	d	dog, dad, milled
46	2e	P*	p	poke, pin, dip
47	2f	B*	b	bad, bug, blubber
48	30	K*	k	kit, cake, folk
49	31	G*	g	gun, ghost, again
50	32	J*	dʒ	jam, wage, edge
51	33	CH	tʃ	chip, speech, future
52	34	_1	|	pause, very short
53	35	_2	|	pause, length of short vowel
54	36	_3	||	pause, length of long vowel
55	37	_4	||	pause, twice long vowel

Phoneme Example

Say we want to encode the word "hello." We can easily do this word just by sounding it out, but for completeness we'll consult a dictionary to find the IPA assignment. Several options may be listed, but we'll use the primary American pronunciation: /hɛˈloʊ/. We can ignore the stress marker (ˈ) and concentrate on the phonetic symbols.

The first two are easy, h=H* and ɛ=EH. The third is slightly trickier because we do have two separate "L" sounds available, but from the chart examples and from the IPA symbol we can determine we want the "L*" token. Finally the oʊ pair is available as a diphthong using token OW. So our final tag list for this word would be "H*EHL*OW" and the data string would be `0x1f 0x0b 0x23 0x00`.

Speech Speed

The speed at which speech is articulated can be controlled with a special command sequence beginning with `0xf4`. Sending this command byte followed by a data byte will immediately change the speech speed. The data byte should be in the range 0-252, with higher values producing faster speech. The default value is 162.

Speech Factors

During speech articulation, the settings of each of the voice oscillators are controlled according to the standard values for each phoneme. There are eight of these values in total, representing the two settings (frequency and amplitude) for each of the four voice oscillators. Each of these eight settings can be adjusted upward or downward from the normal value before being applied, changing the vocal characteristics of the resulting speech. For this purpose there are eight adjustment factors in the range of 0-255. The default value for each factor is 128, which results in no adjustment being applied. Lower values will adjust the setting downward, while higher settings will adjust the setting upward. The factors are controlled with a special command sequence beginning with 0xf5. Sending this command followed by eight data bytes will set the eight factor values accordingly.

Speech Buffer

Desired sounds are stored in a buffer which can hold up to 255 phonemes. Adding phonemes to the buffer is done by writing to a pseudo-register (number 60). Any values written to "register 60" are stored consecutively in the buffer. New writes will append the new data to the end of the current buffer. Values in the buffer will remain until explicitly cleared. See Command Protocol for details.

Speech Configuration

A few steps are needed before activating speech mode. First the registers for the voice oscillators should be set. REV, INV, and STP should be set to zero, and MID should be set to 128, for each voice oscillator. WAV should be set to Sine (000) for VOSC0-VOSC2, and Noise (101) for VOSC3. This is the standard setup for highest-quality speech, but can be altered as desired for various sounds or special effects.

Next, one of the modulation oscillators should be chosen to control the pitch of the vocal output. The desired pitch (in hertz) should be placed in the FRQ register of the chosen oscillator, in the usual format (whole number component in bits 4-15, fractional component in bits 0-3).

Finally a patch must be configured to apply the pitch to VOSC0-VOSC2. For this patch PSC should be set to the chosen modulation oscillator and PDN should be set to zero. Two patch options are available, each producing a slightly different sound. The first option is PTY=13. If MOSC0 is chosen this corresponds to patch bytes `0x0d 0x00` (The PPM bits don't matter for this patch). The other option is PTY=10, which corresponds to patch bytes `0x0a 0xff` (PPM should be 255 for this patch). In this case the waveform for MOSC0 should also be set to Hilltop.

Speech Example

Let's consider a full set of commands to speak the word "hello." The commands will be shown on separate lines with line numbers for clarity and ease of reference, though in actual usage they would simply be a continuous stream of bytes and could be consolidated somewhat.

```
1: 0x00 0x00
2: 0x7d 0xa0
3: 0x02 0x80 0x07
4: 0xca 0x0d 0x00
5: 0x8d 0x50 0x9d 0x50 0xad 0x50 0xbd 0x50
6: 0xc1 0xff
7: 0xf0 0x04 0x1f 0x0b 0x23 0x00
8: 0xf7
```

Let's break this down line-by-line:

Line 1: Reset all registers to default values to start from a known configuration

Line 2: Set VOSC3 waveform to Noise (leave others at default of Sine)

Line 3: Set MOSC0 frequency to 120 Hz (voice pitch)

Line 4: Create a patch of type 13, synchronizing VOSC0-VOSC2 to MOSC0.

Line 5: Set MIX volume to 5 for each voice oscillator

Line 6: Set VOL (overall volume) to maximum and enable all oscillator outputs

Line 7: Load speech buffer with desired phoneme tokens

Line 8: Start speaking

Note that speech automatically stops at the end of the speech buffer, but there is also a "Stop Speaking" command (`0xf6`) in case you want to stop speech mode at an earlier point.

Sleep Mode

If no sound output will be needed for some time, you may want to enter sleep mode. In sleep mode, all output is suspended and the audio amplifier is disabled, significantly reducing power usage.

Entering Sleep Mode

To enter sleep mode, simply send the command byte `0xfb`. It should be sent as a single transaction or as the last byte of a sequence, with no bytes following. Any data arriving while sleep mode is engaging could abort the shut-down operation and potentially leave the input buffer in a confused state.

Exiting Sleep Mode: I^{2}C

When using the I^{2}C interface, sleep mode can be exited simply by beginning a new data transmission. Wake-up will occur automatically once the proper slave address is recognized, and the following data byte(s) will be processed normally with no special adjustments.

Only transmissions beginning with the assigned slave address will trigger wake-up, so other transmissions (to other devices) can still occur on the network without consequence.

Exiting Sleep Mode: SPI

When using the SPI interface, asserting the $\overline{\text{SS}}$ signal (bringing the line low) will exit sleep mode. However, data transmission should not occur immediately, as the wake-up process is not instantaneous and bits will be lost if sent before normal operation is restored.

If possible, wake-up should first be triggered by bringing $\overline{\text{SS}}$ low for around 100μs, then raising it, without transmitting any data. Then normal transmission procedures can resume.

If that is not practical, then the same result can be achieved by transmitting a no-op command (byte `0xff`) as an isolated transaction. That is, $\overline{\text{SS}}$ should be lowered, the byte `0xff` transmitted, and $\overline{\text{SS}}$ raised. After this, wake-up should be completed and normal transmission procedures can resume.