BuzzKill
Sound Effects Board

Arduino Library Guide
version 1.0

Last updated May 12, 2025

Library Overview

This library is intended to provide a simple and logical interface for controlling the BuzzKill board from within an Arduino sketch, without needing to deal with the underlying logistics. The command protocol used by the BuzzKill board is designed for brevity and efficiency, and thus can be tricky for direct human usage. Furthermore, it utilizes a register-based structure where board functions are controlled by setting various register locations to certain values. This again can be tricky for humans to use directly. Therefore this library will provide an abstraction layer with methods that perform intuitive operations such as changing a frequency or starting a note, rather than manipulating binary register values.

This guide only attempts to describe the purpose and usage of the library methods themselves, without delving into the board-level details. Although it is certainly possible to use only the examples given here to achieve basic results, it is highly recommended to read through the User Guide for a fuller understanding of the hardware operation before tackling more advanced methods.

The library methods are divided into these categories: initialization, oscillators, envelopes, patches, speech, output, registers, and miscellaneous. These categories roughly reflect the hardware and firmware implementations of the BuzzKill board. Each category will be described in detail in the following sections.

Finally a simplified summary list of all methods will be provided for quick reference.

Initialization Methods

An initialization method must be called before any other method is used. Initialization prepares a connection at the hardware level, therefore separate initialization methods are used depending on whether the underlying hardware interface is SPI or I2C.

beginSPI

Use this method to initialize a library instance using an SPI interface. In most cases, no arguments will be needed.

Before calling this method, be sure to include the SPI.h library and to call SPI.begin().

If you wish to use a non-default pin for the SPI $\overline{\text{SS}}$ signal, you must specify it as the first (or only) argument.

If your desired SPI library instance is not named “SPI” (typically because you have multiple SPI interfaces with different names), you must specify the appropriate instance as the second argument.

Examples:

```
beginSPI();                           // Initialize library with defaults
beginSPI(12);                         // Initialize with pin 12 as S̅S̅
beginSPI(10, SPI2);                   // Initialize with pin 10 using SPI2
```

beginI2C

Use this method to initialize a library instance using an I2C interface. In most cases, no arguments will be needed.

Before calling this method, be sure to include the Wire.h library and to call Wire.begin().

If you have changed the I2C address of the BuzzKill board, you must specify the appropriate address as the first (or only) argument.

If your desired I2C library instance is not named “Wire” (typically because you have multiple I2C interfaces with different names), you must specify the appropriate instance as the second argument.

Examples:

```
beginI2C();                           // Initialize library with defaults
beginI2C(0x61);                       // Initialize with I2C address 0x61
beginI2C(0x0a, Wire2);                // Initialize with address 0x0a using Wire2
```

Oscillator Methods

Oscillators are divided into two separate types: voice oscillators and modulation oscillators. Each type contains four oscillators, numbered 0-3. Thus each oscillators method begins with an oscillator type constant (to specify voice or modulation) and an oscillator number (0-3) within that type.

setFrequency

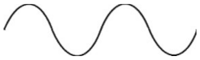
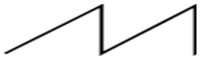
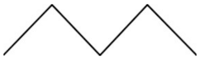
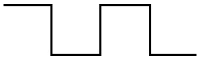
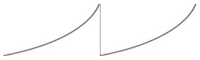
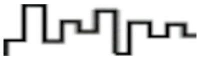
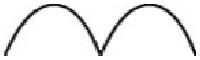
Use this method to set the frequency of an oscillator. Specify the oscillator type and number, along with the desired frequency in hertz. If you want a fractional frequency, use a decimal fraction in steps of 1/16th (e.g. .0625, .125, .1875 etc.). The maximum frequency is 4095.9375.

Examples:

```
setFrequency(BUZZKILL_OSCTYPE_VOICE, 0, 200);    // Voice Osc 0 now 200 Hz
setFrequency(BUZZKILL_OSCTYPE_MOD, 2, 1.375);    // Mod Osc 2 now 1.375 Hz
```

setShape

Use this method to select the waveform shape. Specify the oscillator type and number, along with a shape designator according to the following table.

Designator	*Shape*	*Description*
BUZZKILL_SHAPE_SINE		The most "pure" sound, very smooth and clean with no extra harmonics.
BUZZKILL_SHAPE_RAMP		A "buzzy" sound with many harmonics. Can resemble stringed instruments.
BUZZKILL_SHAPE_TRIANGLE		Some harmonics, smoother than a ramp but harsher than a sine. Can resemble a piano or xylophone.
BUZZKILL_SHAPE_PULSE		Some weaker harmonics. Can sound "electronic" or like reed instruments.
BUZZKILL_SHAPE_EXPON		Similar to a ramp, more useful for modulations than direct audio usage.
BUZZKILL_SHAPE_NOISE		Useful for drums, gunshots, crowds cheering, and wind sounds.
BUZZKILL_SHAPE_HILLTOP		Similar to a triangle, more useful for modulations than direct audio usage.
BUZZKILL_SHAPE_CUSTOM		User defined, so can sound like anything.

Example:

```
setShape(BUZZKILL_OSCTYPE_VOICE, 0, BUZZKILL_SHAPE_PULSE);
```

setMidpoint

Use this method to set the midpoint of an oscillator. Specify the oscillator type and number, along with the desired midpoint in the range 0-255. A value of 128 is the default, and will result in a “normal” shape for most waveforms. For pulse waveforms, this value controls the duty cycle: 64=25%, 128=50%, 192=75%, etc.

Example:

```
setMidpoint(BUZZKILL_OSCTYPE_VOICE, 0, 128);     // Voice Osc 0 midpoint now 50%
```

setInvert

Use this method to invert the output of an oscillator. Specify the oscillator type and number, along with a true/false value. A true value will invert the output (mirror it vertically).

Example:

```
setInvert(BUZZKILL_OSCTYPE_VOICE, 0, true);      // Voice Osc 0 output now inverted
```

setReverse

Use this method to reverse the output of an oscillator. Specify the oscillator type and number, along with a true/false value. A true value will reverse the output (mirror it horizontally).

Example:

```
setReverse(BUZZKILL_OSCTYPE_VOICE, 0, true);     // Voice Osc 0 output now reversed
```

setStep

Use this method to set the step size of an oscillator. Specify the oscillator type and number, along with a value from 0 to 7. The step size represents how many bits to remove from the normally 8-bit output value, so for example at step=4 the output will be restricted to only 16 levels, while at step=7 the output will be restricted to only 2 levels.

Example:

```
setStep(BUZZKILL_OSCTYPE_VOICE, 0, 2);    // Voice Osc 0 output now uses 64 levels
```

configureOscillator

Use this method to completely configure an oscillator in a single call. Specify the oscillator type and number, followed by values for frequency, shape, and optionally midpoint, invert, reverse, and step.

Examples:

```
// Set voice osc 0 to 500 Hz sine wave, normal midpoint, no invert/reverse/step
configureOscillator(BUZZKILL_OSCTYPE_VOICE, 0, 500, BUZZKILL_SHAPE_SINE);
```

```
// Set mod osc 1 to inverted 5 Hz pulse wave with 25% duty cycle
configureOscillator(BUZZKILL_OSCTYPE_MOD, 1, 5, BUZZKILL_SHAPE_PULSE, 64, true);
```

restartOscillators

Use this method to restart one or more oscillators. Specify a bitmask value for which oscillators to restart. The bitmask contains a bit position for each oscillator, with mod oscillators 0-3 in bit positions 0-3 and voice oscillators 0-3 in bit positions 4-7.

A restarted oscillator will immediately return to its starting value and continue its cycle from the beginning. This is useful when oscillators need to be precisely synchronized, or when you need to ensure that a low-frequency modulation starts at a known point.

Example:

```
restartOscillators(0b00010010);              // Restart voice osc 0 and mod osc 1
```

haltOscillators

Use this method to halt one or more oscillators. Specify a bitmask value for which oscillators to halt. The bitmask contains a bit position for each oscillator, with mod oscillators 0-3 in bit positions 0-3 and voice oscillators 0-3 in bit positions 4-7.

A halted oscillator will be held at its starting value. It will remain halted until it is explicitly un-halted, either by another call to this method or a write/reset of the HLT register. This is useful when oscillators need to be precisely synchronized, or when you want oscillators to run only at precise times.

Example:

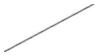
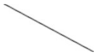
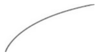
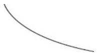
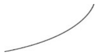
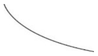
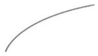
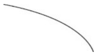
```
haltOscillators(0b00101000);                 // Halt voice osc 1 and mod osc 3
```

Envelope Methods

Envelopes are responsible for applying volume contours to the outputs of the voice oscillators. There are four envelopes, numbered 0-3, to correspond with the four voice oscillators 0-3.

setCurve

Use this method to select an envelope curve type. Specify the envelope number and curve type designator according to the table at right.

Designator	*Attack Shape*	*Decay/Release Shape*
BUZZKILL_CURVE_LINEAR		
BUZZKILL_CURVE_NATURAL		
BUZZKILL_CURVE_INVATT		
BUZZKILL_CURVE_INVDEC		

Example:

```
setCurve(0, BUZZKILL_CURVE_NATURAL);         // Env 0 now using natural curve
```

setAttack

Use this method to set the envelope attack time. First specify the envelope number; you may then specify either the explicit attack range and value, or a single value representing the desired time in ms (in which case the closest range/value pair will be automatically calculated and set).

Examples:

```
setAttack(0, 1, 4);                          // Env 0 attack time now 205 ms
setAttack(1, 250);                           // Env 1 attack time now 246 ms
```

setDecay

Use this method to set the envelope decay time. First specify the envelope number; you may then specify either the explicit decay range and value, or a single value representing the desired time in ms (in which case the closest range/value pair will be automatically calculated and set).

Examples:

```
setDecay(0, 1, 4);                           // Env 0 decay time now 205 ms
setDecay(1, 250);                            // Env 1 decay time now 246 ms
```

setSustain

Use this method to set the envelope sustain level. Specify the envelope number and the desired sustain level in the range 0-127 (0=no sustain, 127=sustain at 100%).

Example:

```
setSustain(0, 64);                           // Env 0 sustain level now at 50%
```

setRelease

Use this method to set the envelope release time. First specify the envelope number; you may then specify either the explicit release range and value, or a single value representing the desired time in ms (in which case the closest range/value pair will be automatically calculated and set).

Examples:

```
setRelease(0, 1, 4);                         // Env 0 release time now 205 ms
setRelease(1, 250);                          // Env 1 release time now 246 ms
```

noteOn

Use this method to control the envelope gate, starting or stopping a note. This method may be utilized in two distinct ways:

You may specify an envelope number and an optional true/false value. If the value is true (or absent) the gate will be enabled, starting a note. If false the gate will be disabled, stopping a note.

Alternatively you may specify four separate true/false values, which will be applied to envelopes 0-3 in order. For each envelope a value of true will activate the gate (start a note), while a value of false will deactivate it (stop a note).

Examples:

```
noteOn(0);                                   // Env 0 gate on (start note)
noteOn(1, false);                            // Env 1 gate off (stop note)
noteOn(true, true, true, false);             // Env 0-2 gate on, env 3 gate off
```

noteOff

Use this method to disable the envelope gate, stopping a note. Specify the envelope number. Calling noteOff(x) is equivalent to calling noteOn(x, false).

Example:

```
noteOff(0);                                  // Env 0 gate now off (note stopped)
```

setMixVolume

Use this method to set the envelope mix volume level. Specify the envelope number and the desired mix volume in the range 0-15. This setting controls the “loudness” of the envelope relative to other envelopes. A good rule-of-thumb is to have the mix volumes of all enabled voices total to 14 or 15.

Example:

```
setMixVolume(0, 15);                         // Env 0 now at maximum mix level
```

configureEnvelope

Use this method to completely configure an envelope in a single call. Specify the envelope number, the curve type, the attack time, the decay time, the sustain level, the release time, the mix volume, and the gate status. The attack/decay/release times may be specified as single values representing the nearest time in milliseconds, or as paired values representing the desired range/rate combination.

Examples:

```
// Config env 0 with linear curve, attack 328ms, decay 287ms, sustain 50%,
//   release 410ms, mix volume 15, and gate enabled (note on)
configureEnvelope(0, BUZZKILL_CURVE_LINEAR, 1, 10, 1, 8, 64, 1, 14, 15, true);

// Same as above but using simplified time parameters
configureEnvelope(0, BUZZKILL_CURVE_LINEAR, 328, 287, 64, 410, 15, true);
```

Patch Methods

addPatch

Use this method to install a new modulation patch. Specify the source (mod) oscillator, the voice (destination) oscillator, the patch type designator (see table below), and the patch parameter. This method will return a value representing the slot number (0-4) to which the new patch was assigned, or 255 if no slot is available. This value should be stored in case the patch later needs to be removed.

Designator	*Description*	*Parameter Use*
BUZZKILL_PATCH_NONE	None (disabled)	N/A
BUZZKILL_PATCH_FREQSCALE	Frequency Scale	Scaling Factor (1-255)
BUZZKILL_PATCH_FREQSHIFT	Frequency Shift	Scaling Factor (1-255)
BUZZKILL_PATCH_MIDSHIFT	Midpoint Shift	Scaling Factor (1-255)
BUZZKILL_PATCH_AMPSCALE	Amplitude Scale	Scaling Factor (1-255)
BUZZKILL_PATCH_AMPLEVEL	Amplitude Level	Scaling Factor (±1-127)
BUZZKILL_PATCH_ENVGATE	Envelope Gate	Threshold (1-255)
BUZZKILL_PATCH_HARDSYNCH	Hard Synch	N/A
BUZZKILL_PATCH_SOFTSYNCH	Soft Synch	Bitfield: -- -- -- RV IN -- -- RS
BUZZKILL_PATCH_RINGMOD	Ring Modulation	N/A
BUZZKILL_PATCH_AMPSCALEMULTI	Amplitude Scale Multi	Scaling Factor (1-255)
BUZZKILL_PATCH_AMPLEVELMULTI	Amplitude Level Multi	Scaling Factor (±1-127)
BUZZKILL_PATCH_ENVGATEMULTI	Envelope Gate Multi	Threshold (1-255)
BUZZKILL_PATCH_HARDSYNCHMULTI	Hard Synch Multi	N/A
BUZZKILL_PATCH_SOFTSYNCHMULTI	Soft Synch Multi	Bitfield: -- -- -- RV IN -- -- RS
BUZZKILL_PATCH_OUTPUTPIN	Output Pin	Threshold (1-255)

Example:

```
slot=addPatch(0, 1, BUZZKILL_PATCH_FREQSHIFT, 50);  // Patch from mosc0 to vosc1
```

removePatch

Use this method to remove a previously added patch. Specify the slot number assigned when the patch was added.

Example:

```
removePatch(0);                                     // Remove the patch in slot 0
```

clearPatches

Use this method to remove all patches at once. No arguments are required.

Example:

```
clearPatches();                                     // All patches now inactive
```

Speech Methods

addSpeechPhonemes

Use this method to add new phonemes to the speech buffer. The new phonemes will be appended to any currently in the buffer. Specify either a byte array, a char array, or a string literal containing the desired phoneme values. A length argument may optionally be specified, otherwise the array or string must be terminated by a 255 (0xff) value.

Examples:

```
byte barr[]={31, 11, 35, 0}; addSpeechPhonemes(barr, 4);       // Add with length
char carr[]={31, 11, 35, 0, 255}; addSpeechPhonemes(carr);     // Implied length
addSpeechPhonemes(“\x1f\x0b\x23\x00\xff”);                      // String literal
```

addSpeechTags

Use this method to add new phonemes to the speech buffer. The new phonemes will be appended to any currently in the buffer. Specify either a char array, a C-style string, or a string literal containing a sequence of two-letter text tags representing the desired phonemes. Spaces may be optionally included between tags for clarity. A length argument may be specified (representing the number of phonemes, not characters). Otherwise an array must be terminated with either a ‘.’ character or a null character (this is automatic for string literals).

Examples:

```
char carr[]=”H* EH L* OW”; addSpeechTags(carr, 4);      // Add with explicit length
addSpeechTags(“H* EH L* OW”);                            // Add with implied length
```

getPhonemeFromTag

Use this method to convert a two-letter tag into the corresponding phoneme value. Specify a char array containing at least two characters representing the desired phoneme. It returns a value 0-55.

Example:

```
x = getPhonemeFromTag(“Z*”);                             // x = 24
```

clearSpeechBuffer

Use this method to clear the speech buffer of all phonemes. No argument is required. This is the only way to remove phonemes from the speech buffer once they have been added.

Example:

```
clearSpeechBuffer();                                     // Speech buffer now empty
```

setSpeechSpeed

Use this method to set the speed at which speech is rendered. Specify a speed in the range 0-252. Higher values produce faster speech. The default value is 162.

Example:

```
setSpeechSpeed(100);                                     // Speak at slow rate
```

setSpeechFactors

Use this method to set the speech adjustment factors, which scale the voice frequency and amplitude components upward or downward to alter the vocal characteristics. There are separate adjustment factors for the frequency and amplitude components for each of the four vocal formants. Specify the desired factors as frequency/amplitude pairs for formant 1, then formant 2, etc.

Example:

```
setSpeechFactors(55, 128, 182, 128, 199, 128, 255, 128);   // Restore defaults
```

prepareSpeechMode

Use this method to quickly set all oscillators, patches, and mix volumes to appropriate values for producing speech. Specify a pitch value in hertz, and a patch type for applying the pitch. The patch type should typically be one of either AMPSCALEMULTI or HARDSYNCHMULTI.

Examples:

```
prepareSpeechMode(150, BUZZKILL_PATCH_AMPSCALEMULTI);   // 150Hz with amp patch
prepareSpeechMode(160, BUZZKILL_PATCH_HARDSYNCHMULTI);  // 160Hz with synch patch
```

startSpeaking

Use this method to begin speaking the phonemes in the speech buffer. No argument is required. Speaking will continue until the end of the buffer is reached, or until manually stopped.

Example:

```
startSpeaking();                                        // Begin speaking
```

stopSpeaking

Use this method to stop speech currently in progress. No argument is required. This method is only needed if you wish to interrupt speaking before the end of the buffer is reached.

Example:

```
stopSpeaking();                                         // Stop any ongoing speech
```

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*	ɦ	ahead, behind, Hanukkah
33	21	R*	ɹ	red, carrot, wrench
34	22	RX	ɹ̩	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

Output Methods

The final output of each voice oscillator, after processing by its associated envelope generator and its mix volume level, is termed a “voice.” Voice 0 represents the final output of voice oscillator 0, Voice 1 is the final output of voice oscillator 1, etc. Each voice can be independently enabled or disabled. All enabled voices are then combined, adjusted by the master volume level, and output to the speaker.

enableVoice

Use this method to control the audible output from a voice oscillator. This method may be utilized in two distinct ways:

You may specify a voice number and an optional true/false value. If the value is true (or absent) the output will be enabled, becoming audible from the speaker. If false, the output will be disabled and no longer audible.

Alternatively you may specify four separate true/false values, which will be applied to voices 0-3 in order. For each voice a value of true will activate the output, while a value of false will deactivate it.

Examples:

```
enableVoice(0);                                 // Voice 0 now audible
enableVoice(1, false);                          // Voice 1 now inaudible
enableVoice(true, true, true, false);           // Voice 0-2 audible, 3 inaudible
```

disableVoice

Use this method to disable a voice output, making it inaudible. Specify the voice number. Calling disableVoice(x) is equivalent to calling enableVoice(x, false).

Example:

```
disableVoice(0);                                // Voice 0 now inaudible
```

setMasterVolume

Use this method to set the master volume level for all sound output. Specify the volume level in the range of 0-15. 0 represents the minimum possible volume level, but will still produce output. If silence is desired, be sure to disable all voices so no output is produced.

Example:

```
setMasterVolume(15);                            // Set master volume to max
```

Register Methods

resetRegisters

Use this method to reset some or all registers to their default values. This will disable all sound output and clear all patches, and is useful for quickly returning the board to a known state. You may optionally specify a starting register number (0-59), in which case only the registers from that point forward will be reset.

Resetting registers does affect speech settings or clear the speech buffer.

Examples:

```
resetRegisters();                          // Reset all registers
resetRegisters(50);                        // Reset registers 50-59 (clears patches)
```

The following two methods for manually setting register values are generally redundant, but are provided for completeness. In some instances it may be more efficient to set a number of registers directly than to call the corresponding specialty methods. In most code however, users should favor the special-purpose methods outlined previously.

setRegister

Use this method to directly set the value of a board register. Specify the register number (0-59), followed by the desired value. You may optionally include additional values, which will be written to subsequent registers in order. You may set the values of up to 10 consecutive registers this way.

Examples:

```
setRegister(18, 0xc0);                          // Set board register 18 to 0xc0
setRegister(16, 0xe0, 0x80, 0xc0, 0x83);        // Set values of registers 16-19
```

writeRegisters

Use this method to transfer values from a byte array, char array, or string literal to a set of consecutive board registers. Specify the starting register number (0-59), followed by the data source, and then a length argument.

Examples:

```
byte barr[]={100, 55, 72, 15}; writeRegisters(0, barr, 4);     // Set registers 0-3
char carr[]={200, 78, 18, 43}; writeRegisters(4, carr, 4);     // Set registers 4-7
writeRegisters(2, “\xe0\x80\xc0\x83\xf0”, 5);                  // Set registers 2-6
```

Miscellaneous Methods

boardSleep

Use this method to place the board into sleep mode. No argument is required. All output will cease and no power will be sent to the speaker. Power consumption will be minimized, so intelligent use of this mode may greatly increase the run-time of battery-powered projects.

Example:

```
boardSleep();                                         // Enter low-power sleep mode
```

boardWake

Use this method to wake a board which has previously been put into sleep mode. No argument is required. The speaker will be powered and audio output will resume. All registers and settings will retain their previous values.

Example:

```
boardWake();                                          // Exit low-power sleep mode
```

storeCustomWave

Use this method to define a custom waveform shape. Specify a byte array containing at least 256 values, each in the range 0-255. If the array is longer than 256 bytes, only the first 256 will be used.

Example:

```
storeCustomWave(wavedata);                            // wavedata is a byte array
```

changeI2CAddress

Use this method to change the board's I2C address. Specify the new 7-bit address in the range 8-119. This change is permanent (unless changed again) and the new address must be used for all future connections.

You do not need to immediately call beginI2C() again, but the next time you do you must use the new address.

Simplified Method Summary

Brackets indicate optional parameters or parameter groups.

Vertical bars (|) indicate a choice between two possible parameters or parameter groups, each enclosed in brackets. One of the two options must be used, but not both.

Ellipses (...) indicate an option that may be repeated one or more times.

Question marks indicate a boolean value. Argument types are otherwise omitted for simplicity.

Initialization Methods

```
beginSPI([ssPin [, spiInstance]])
beginI2C([address [, wireInstance]])
```

Oscillator Methods

```
setFrequency(oscType, oscNum, frequency)
setMidpoint(oscType, oscNum, midpoint)
setShape(oscType, oscNum, shape)
setInvert(oscType, oscNum, invert?)
setReverse(oscType, oscNum, reverse?)
setStep(oscType, oscNum, step)
configureOscillator(oscType, oscNum, freq, shape [, midpoint, inv?, rev?, step])
restartOscillators(bitmask)
haltOscillators(bitmask)
```

Envelope Methods

```
setCurve(envNum, curveType)
setAttack(envNum, [attackRange, attackVal] | [attackTime])
setDecay(envNum, [decayRange, decayVal] | [decayTime])
setSustain(envNum, sustainLevel)
setRelease(envNum, [releaseRange, releaseVal] | [releaseTime])
setMixVolume(envNum, mixVol)
noteOn(envNum [, gate?])
noteOn(gate0?, gate1?, gate2?, gate3?)
noteOff(envNum)
configureEnvelope(envNum, curveType, attRange, attVal, decRange, decVal, susLevel,
                  relRange, relVal, mixVol, noteOn?)
configureEnvelope(envNum, curveType, attTime, decTime, decTime, susLevel,
                  relTime, mixVol, noteOn?)
```

Patch Methods

```
addPatch(sourceModOsc, destVoiceOsc, patchType, patchParam)
removePatch(patchSlot)
clearPatches()
```

Speech Methods

```
addSpeechPhonemes(phonemeArray [, length])
addSpeechTags(tagStringOrArray [, numOfPhonemes])
getPhonemeFromTag(tagStringOrArray)
clearSpeechBuffer()
setSpeechSpeed(speed)
setSpeechFactors(f1freq, f1amp, f2freq, f2amp, f3freq, f3amp, f4freq, f4amp)
prepareSpeechMode(pitch, patchType)
startSpeaking()
stopSpeaking()
```

Output Methods

```
setMasterVolume(volume)
enableVoice(voiceNum [, enable?)
enableVoice(enable0?, enable1?, enable2?, enable3?)
disableVoice(voiceNum)
```

Register Methods

```
resetRegisters([regStart])
setRegister(regNum, value [, nextValue]...)
writeRegisters(regStart, valueStringOrArray, length)
```

Miscellaneous Methods

```
boardSleep()
boardWake()
storeCustomWave(arrayOf256Bytes)
changeI2CAddress(newAddress)
```